

مقدمة قصيرة جدًّا

سوبراتا داسجوبتا

علم الكمبيوتر

ترجمة إبراهيم سند أحمد

علم الكمبيوتر

مقدمة قصيرة جدًّا

تأليف
سوبراتا داسجوبتا

ترجمة
إبراهيم سند أحمد

مراجعة
عبد الفتاح عبد الله



علم الكمبيوتر

Computer Science

سوبراتا داسجوبتا

Subrata Dasgupta

الناشر مؤسسة هنداوي
المشهرة برقم ١٠٥٨٥٩٧٠ بتاريخ ٢٦ / ١ / ٢٠١٧

يورك هاوس، شييت ستريت، وندسور، SL4 1DD، المملكة المتحدة
تليفون: ‎+ ٤٤ (٠) ١٧٥٣ ٨٣٢٥٢٢
البريد الإلكتروني: hindawi@hindawi.org
الموقع الإلكتروني: https://www.hindawi.org

تصميم الغلاف: ولاء الشاهد

الترقيم الدولي: ٩٧٨ ١ ٥٢٧٣ ٣١٢٩ ٧

صدر الكتاب الأصلي باللغة الإنجليزية عام ٢٠١٦.
صدرت هذه الترجمة عن مؤسسة هنداوي عام ٢٠٢٣.

المحتويات



إهداء إلى أنطون ريبون

شكر وتقدير

أتقدَّم بالشكر إلى لاثا مينون، محرِّرتي لدى مطبعة جامعة أكسفورد، على دعمها ومشوراتها الحكيمة بشأن هذا المشروع منذ بدايته. فقد كانت تعليقاتها على النسخة قبل الأخيرة تعليقاتٍ مفيدة بنحو خاص.

ولم تتوانَ جيني نوجي في الرد بشأن الاقتراحات والمساعدات التحريرية في مختلف مراحل هذا العمل. وإني لأتقدَّم بالشكر إليها.

اطَّلع أربعة قراء مجهولون على مسوَّدتين مختلفتين لمتن الكتاب، وقدَّموا اقتراحات وتعليقات قيمة وقد أخذتها على محمل الجِد. إنني ممتن لهم كلَّ الامتنان، ووددِت لو أني تقدمت بالشكر إليهم بأسمائهم.

وأتوجَّه بالشكر إلى إلمان باشار لإعداد الرسوم التوضيحية.

وقد استُخدمت أجزاء من هذه المادة في منهج دراسي حول «التفكير الحوسبي» ذي مستوًى متقدِّم مقدَّم لطلاب جامعيين، درَّسته في عدة مناسبات لغير المتخصِّصين في علم الكمبيوتر. وكانت استجاباتهم مفيدةً للغاية في رسم ملامح النص وتنقيحه.

وفي النهاية كما هو معتاد دومًا، أتوجَّه بالشكر إلى أفراد عائلتي. ما فتئ كل فرد منهم يقدِّم الدعم — على اختلاف طرقهم — ما يجعل عيش الحياة الفكرية أجدى وأجدر.

المقدمة

كانت ستينيات القرن العشرين فترةً عصيبةً على الصعيدين الاجتماعي والثقافي. لكن بين ثنايا الحرب الباردة وحركة الدفاع عن الحقوق المدنية والمظاهرات المناهضة للحروب والحركة النسوية والثورات الطلابية وشعارات المقاومة السلبية واللاعنف والاعتصامات والتمردُّات اليسارية الراديكالية، وعلى نحوٍ لم يكد ينتبه له أحد، ظهر علمٌ جديد في أحرام الجامعات في الغرب وحتى في بعض الأقطار غير الغربية، وإن كان على نحوٍ أقل وضوحًا.

تركَّز هذا العلم على نوعٍ جديد من الأجهزة؛ ألا وهو جهاز الكمبيوتر الرقمي الإلكتروني. وأُطلق على التكنولوجيا التي تعلَّقت بهذا الجهاز عدة أسماء، والتي من أشيعها «الحوسبة التلقائية» و«معالجة المعلومات». وفي البلدان الناطقة بالإنجليزية، كان يُطلق على نطاق واسع على هذا العلم اسم computer science ويعني «علم الكمبيوتر»، أما في أوروبا فكانت تُستخدم أسماء مثل informatique أو informatik وتعني «المعلوماتية».

تعني الحوسبة التلقائية تصميمَ وصناعة أجهزة قادرة على الحوسبة بأدنى تدخُّل من الإنسان، وترجع «فكرتها التكنولوجية» على الأقل إلى أحلامٍ استحوذت على عقل عالِم الرياضيات والمفكر الإنجليزي تشارلز بابدج في أوائل القرن التاسع عشر، هذا إن لم يكن لها تاريخ أقدَم من ذلك. وقد دُرس «المفهوم الرياضي» للحوسبة لأول مرة في أواخر ثلاثينيات القرن العشرين على يد عالِمَي المنطق آلان تورنج الإنجليزي وألونزو تشرتش الأمريكي. لكن الدافع اللازم لإنشاء «علم تجريبي» ملائم للحوسبة كان عليه الانتظار حتى اختراع كمبيوتر رقمي إلكتروني وتصميمه وتنفيذه في أربعينيات القرن العشرين؛ أي بعد نهاية الحرب العالمية الثانية مباشرةً. وحتى في ذلك الحين، مرَّ هذا العلم بفترة تطوير. فلم يبرُز علمٌ مستقل له اسمه وهويته الخاصة إلا في ستينيات القرن العشرين عندما بدأت الجامعات

في تقديم درجات علمية للخريجين وطلاب الدراسات العليا في علم الكمبيوتر، وخرج أول جيل من «علماء الكمبيوتر» المدرَّبين رسميًّا من أحرام الجامعات.

منذ اختراع الكمبيوتر الرقمي الإلكتروني عام ١٩٤٦، لا يخفى على أحد النمو المذهل في التكنولوجيات المرتبطة بهذا الجهاز (والتي يُطلق عليها بوجه عام في الوقت الراهن مصطلح «تكنولوجيا المعلومات») وكذلك التحوُّل الثقافي والاجتماعي المصاحب لذلك (الذي يعبَّر عنه بوجه عام بمصطلحات مثل «عصر المعلومات» و«ثورة المعلومات» و«مجتمع المعلومات»). لقد اجتاحتنا فعليًّا هذه البيئة التكنولوجية الاجتماعية. وعلى الرغم من ذلك، فإن «العلم» — المنهج الفكري — الذي تقوم عليه هذه التكنولوجيا يُعدُّ أقلَّ وضوحًا، وبالتأكيد تقل المعرفة به وفهْمه خارج الأوساط العلمية لعلم الكمبيوتر. لكن لا شك أن علم الكمبيوتر يصطف إلى جانب علوم أخرى مثل علم الأحياء الجزيئي والعلوم المعرفية باعتباره من أهم العلوم التي ظهرت بعد الحرب العالمية الثانية. إضافة إلى ذلك، فإن علم الكمبيوتر تكتنفه بعض الغرابة التي تجذب الانتباه إليه وتميزه عن باقي العلوم الأخرى.

في هذا الكتاب، أهدُف إلى أن أقدِّم للقارئ صاحبِ الفضول الفكري والاهتمام الجاد بالأفكار والمبادئ العلمية الأساسَ لفهْم الطبيعة الجوهرية لعلم الكمبيوتر؛ وإن شئت القول، أود أن أثري الفهم العام لهذا العلم الغريب والفريد تاريخيًّا والأكثر تبعات والذي لا يزال حديثًا. بكلمات بسيطة، يسعى هذا الكتاب إلى الإجابة عن السؤال التالي إجابةً مباشرة وفورية ومختصرة: «ما علم الكمبيوتر؟»

قبل أن نبدأ، جديرٌ بنا أن نبدأ بتوضيح بعض المصطلحات. في هذا الكتاب، سأستخدم كلمة «حوسبة» للإشارة إلى نوع معيَّن من الأنشطة وفي نفس الوقت إلى ناتجه، وبالتبعية كلمة «حوسبي» ستكون صفة في هذا الإطار؛ و«كمبيوتر» للإشارة إلى الجهاز أو الأداة أو النظام الذي يُجري عملية الحوسبة؛ و«أداة» للإشارة إلى أي شيء من صنع الإنسان (أو الحيوان في بعض الأحيان)؛ و«أداة حوسبية» للإشارة إلى أي أداة داخلة في المهام الحوسبية.

وأخيرًا، لا بد من ذكر تحذير. يبدأ هذا الكتاب بقبول الافتراض القائل بأن علم الكمبيوتر هو علمٌ فعليًّا؛ بمعنى أنه يجسِّد السِّمات العامة المرتبطة بمفهوم العلم، ولا سيما أنه يستلزم مزيجًا منهجيًّا من الأساليب التجريبية والمفاهيمية والرياضية والمنطقية والكمية والنوعية للتحقيق والبحث في طبيعة نوع معيَّن من الظواهر. وتقصي هذا الافتراض جزء من فلسفة العلم ولا يدخل في نطاق هذا الكتاب. القضية الأساسية التي تهمنا هنا هي «طبيعة» علم الكمبيوتر «بصفته» علمًا، ولا سيما طابعه المستقل والمميز.

الفصل الأول

«مقومات» الحوسبة

ما علم الكمبيوتر؟ في عام ١٩٦٧، طرح ثلاثة من أوائل المساهمين البارزين في هذا العلم وهم آلان بيرليس وألين نيويل وهيربرت سايمون إجابةً عن هذا السؤال، غير أنها باتت كلاسيكية الآن، وتقول الإجابة ببساطة إن علم الكمبيوتر هو دراسة أجهزة الكمبيوتر والظواهر المرتبطة بها.

هذه إجابة مباشرة تمامًا وأعتقد أن معظم علماء الكمبيوتر سيقبلونها بمثابة تعريف تقريبي يفي بالغرض. يرتكز هذا التعريف على جهاز الكمبيوتر نفسه، وبالتأكيد لن يوجد علم الكمبيوتر من دون جهاز الكمبيوتر. لكن ربما يرغب كلٌّ من المتخصصين في علم الكمبيوتر وغير المتخصصين في فهْم العبارتين المذكورتين في هذا التعريف فهمًا أدقَّ وهما: «أجهزة الكمبيوتر» و«الظواهر المرتبطة بها».

أوتوماتون اسمه «الكمبيوتر»

الكمبيوتر عبارة عن «أوتوماتون». في الماضي كانت تلك الكلمة — والتي صيغت في القرن السابع عشر (وتُجمع على «أوتوماتا») — تعني أي أداة تؤدي أنماطًا مكررة من الحركات والأفعال من دون تأثيرات خارجية، ويدفعها إلى حد كبير مصدرها الخاص من الطاقة المحركة. في بعض الأحيان، كانت تحاكي هذه الأفعال أفعالَ البشر والحيوانات. ابتُكرت هذه الأدوات الميكانيكية الشهيرة منذ العصور القديمة قبل المسيحية، وكان جُل استخدامها لتسلية الأثرياء، ولكن كان لبعضها استخدامات عملية مثل الساعة المائية التي يقال إنها اختُرعت في القرن الأول الميلادي على يد المهندس هيرو السكندري. وتُعدُّ الساعة الميكانيكية ذات الثقالة التي اختُرعت في إيطاليا في القرن الخامس عشر سليلًا ناجحًا وباقيًا لهذا النوع

من الأدوات. في الثورة الصناعية التي قامت في القرن الثامن عشر، كان تشغيل مضخة لشفط المياه من المناجم باستخدام محرك يعمل بالبخار «الجوي» والذي اخترعه توماس نيوكومن (عام ١٧١٣) وطوَّره بعد ذلك جيمس وات وآخرون (عام ١٧٦٥) يُعَد مثالًا آخر على مثل هذه الأدوات العملية.

من ثَم نجد أن الأوتوماتا الميكانيكية التي تؤدي بعض الأعمال الفعلية من نوع أو آخر لها تاريخ زاخر. أما الأوتوماتا التي تحاكي الأفعال الإدراكية فلها تاريخٌ أحدث بكثير. من الأمثلة البارزة على ذلك الروبوت السلحفاة «ماكينا سبيكيولاتريكس» (أي، «الآلة المفكِّرة») الذي اخترعه عالِم الفسيولوجيا العصبية البريطاني ويليام جراي والتر في أواخر أربعينيات وأوائل خمسينيات القرن العشرين. لكن الكمبيوتر الرقمي الإلكتروني التلقائي الذي طُوِّر في النصف الثاني من أربعينيات القرن العشرين كان علامة على ميلاد جنس جديد تمامًا من الأوتوماتا؛ ذلك أن الكمبيوتر كان أداة صُمِّمت لمحاكاة وتقليد أنواع معينة من عمليات «التفكير» البشرية.

إن فكرة الحوسبة باعتبارها طريقةً لمحاكاة التفكير الإنساني — أو الكمبيوتر باعتباره «جهازًا مفكِّرًا» — لهي فكرة مثيرة للاهتمام والحَيرة والجدل، وسأتناولها لاحقًا في هذا الكتاب؛ حيث إنها الأصل في أحد فروع علم الكمبيوتر الذي يسمى «الذكاء الاصطناعي». لكن يميل العديد من علماء الكمبيوتر إلى تقليل الارتباط بالإنسان عند الحديث عن علمهم. حتى إن بعضهم ينفي أيَّ صلة للحوسبة بالتفكير الإنساني المستقل. أشارت عالمة الرياضيات البارزة آدا أوجستس، ذات الأصل الإنجليزي وكونتيسة لوفليس وزميلة تشارلز بابدج (ارجع إلى المقدمة)، في أربعينيات القرن التاسع عشر، إلى أن الجهاز الذي تصوَّره بابدج (يسمَّى المحرِّك التحليلي، وكان أول تجسيد لما عُرف بعد قرن من الزمان باسم الكمبيوتر الرقمي المتعدد الأغراض الحديث) لا يدَّعي بدءَ المهام من تلقاء نفسه. فهو لا ينجز سوى الأوامر التي يمليها عليه الإنسان. وكثيرًا ما يتكرر هذا الرأي لدى المشككين العصريين في الذكاء الاصطناعي، ومنهم السير موريس ويلكس، وهو أحد الرواد في مجال الكمبيوتر الإلكتروني. فقد أكَّد في نهاية القرن العشرين مردِّدًا ما أشارت إليه كونتيسة لوفليس أن أجهزة الكمبيوتر «لا تنجز سوى المهام التي تُملى عليها».

إذن، ما الذي تنجزه أجهزة الكمبيوتر ويميزها عن باقي أنواع الأدوات، بما فيها الأنواع الأخرى من الأوتوماتا؟ وما الذي يجعل علم الكمبيوتر مجالًا علميًّا مميزًا؟

تحقيقًا لهدف هذا الفصل، سأتعامل مع جهاز الكمبيوتر باعتباره «صندوقًا أسود». بمعنى أننا سنتجاهل بشكلٍ أو بآخر البنيةَ الداخلية لأجهزة الكمبيوتر وطريقة عملها؛

حيث سنتطرق إلى ذلك لاحقًا. أما في الوقت الحالي، فسنعتبر جهاز الكمبيوتر نوعًا عامًّا من الأوتوماتون، وسنتناول «المهام» التي ينجزها وليس «الكيفية» التي ينجزها بها.

الحوسبة كمعالجة للمعلومات

كل مجال يطمح إلى أن يكون «علميًّا» يتقيد بـ «المقومات» الأساسية المعنيِّ بها. تتمثل مقوِّمات علم الفيزياء في المادة والقوة والطاقة والحركة؛ وتتمثل مقوِّمات علم الكيمياء في الذرات والجزيئات؛ وتتمثل مقوِّمات علم الوراثة في الجينات؛ وتتمثل مقوِّمات الهندسة المدنية في القوى التي تحافظ على توازن المنشآت المادية.

والرأي السائد بين علماء الكمبيوتر هو أن المقوِّم الأساسي لعلم الكمبيوتر هو «المعلومات». ومن ثَم جهاز الكمبيوتر عبارة عن وسيلة تُستخدم لاسترداد المعلومات من «البيئة» ثم تخزينها أو معالجتها أو تحويلها، ثم إعادة إطلاقها في تلك البيئة، وذلك بشكل تلقائي. وهذا يفسِّر استخدامَ مصطلح بديل للحوسبة وهو «معالجة المعلومات»؛ ويفسِّر الإشارةَ إلى علم الكمبيوتر في أوروبا بمصطلح «المعلوماتية»؛ ويفسِّر سببَ تسمية المنظَّمة المعنية بعمليات الحوسبة باسم الاتحاد الدولي لمعالجة المعلومات، والتي تشبه الأمم المتحدة في مجالها.

مكمن المشكلة هو أنه على الرغم من تأسيس الاتحاد الدولي لمعالجة المعلومات عام ١٩٦٠ (ما يعطي مباركة دولية رسمية لمصطلح معالجة المعلومات)، فإنه لا يزال ثمة قدْر كبير من سوء الفهْم بشأن ماهية المعلومات حتى يومنا هذا. وكما قال موريس ويلكس ذات مرة، إنها شيء «مراوغ».

المعلومات «غير الدلالية»

من أهم أسباب سوء الفهم هذا هو الحقيقة المؤسفة التي مُفادها أن مهندسي الاتصالات استخدموا كلمة «معلومات» بمعنًى مختلفٍ تمامًا عن معناها الدارج. عادةً ما نعني بالمعلومات أنها شيء يخبرنا بشيء عن العالَم من حولنا. في اللغة العادية، المعلومات شيء «له دلالة». فالجملة «متوسط درجة حرارة الشتاء في البلد الفلاني هو خمس درجات مئوية» تخبرنا شيئًا عن مناخ ذلك البلد؛ إنها تعطينا معلومة عن ذلك البلد. في المقابل، في فرع هندسة الاتصالات المسمى «نظرية المعلومات»، والتي ابتكر معظمها مهندس الكهرباء

الأمريكي كلود شانون عام ١٩٤٨، تكون المعلومات ببساطة سلعة تنتقل عبْر قنوات الاتصال مثل أسلاك التلغراف وخطوط الهاتف. في نظرية المعلومات، تكون المعلومات مجردة من المعنى. ويطلق على الوحدة الأساسية للمعلومات في نظرية المعلومات اسم «البت» (بالإنجليزية Bit وهي اختصار للمصطلح binary digit وتعني الرقم الثنائي) ووحدة البت محصورة بين قيمتين، هما في العموم 0 و1. لكن في هذا العصر الذي كثرت فيه أجهزة الكمبيوتر الشخصية والمحمولة، زادت معرفة الناس بمفهوم «البايت». تتكوَّن وحدة البايت الواحدة من ثماني وحدات بت. وبما أن كل وحدة بت يمكن أن تحتوي على قيمة واحدة من قيمتين، فإن وحدة البايت من المعلومات يمكن أن تحتوي على $2^8(256)$ قيمة محتملة تتراوح ما بين 00000000 إلى 11111111. ودلالة وحدات البت (أو البايت) ليست ذات أهمية في هذا المعنى لـ «المعلومات».

في الحوسبة، لا شك أن معالجة المعلومات بهذا المعنى غير الدلالي لها أهمية؛ حيث (كما سنرى) إن جهاز الكمبيوتر المادي — المصنوع من دوائر إلكترونية وأجهزة مغناطيسية وكهروميكانيكية وغيرها (والتي يطلق عليها مجتمعة اسم «العتاد») — يخزن المعلومات ويعالجها وينقلها في صورة مضاعفات من وحدات البت والبايت. في الحقيقة، مفهوم وحدات البت والبايت من الطرق التي تُحدَّد بها قدرات الأداة الحوسبية وأداؤها. على سبيل المثال، يمكن أن أشتري جهاز كمبيوتر محمول يحتوي على ذاكرة داخلية بسعة 6 جيجابايت وذاكرة خارجية (القرص الصلب) بسعة 500 جيجابايت، (حيث إن 1 جيجابايت = 10^9 بايت)، أو يمكنني التحدُّث عن شبكة أجهزة كمبيوتر تنقل المعلومات بمعدل 100 ميجابت في الثانية (حيث إن 1 ميجابت = 10^3 بت).

المعلومات «الدلالية» (أو ذات المعنى)

لكن جهاز الكمبيوتر المادي ليس سوى نوع واحد من الأدوات الحوسبية (كما سنرى في الفصل الثاني). والمعلومات غير الدلالية ليست سوى نوع واحد من أنواع المعلومات التي يُعنى بها علماء الكمبيوتر. فالنوع الآخر الأكثر أهمية (وربما إثارة للاهتمام) هو المعلومات التي لها دلالة: أي المعلومات «ذات المعنى». ترتبط هذه المعلومات بـ «العالم الواقعي»، وهذا المعنى يتطابق مع الاستخدام الدارج للكلمة. على سبيل المثال، عندما أدخل على شبكة الإنترنت من جهاز الكمبيوتر الشخصي لديَّ، فبالتأكيد أن معالجة المعلومات تحدُث على مستوًى مادي أو «غير دلالي»؛ بمعنى أن المعلومات تُنقل من جهاز كمبيوتر

بعيد (الخادم) عبْر الشبكة إلى جهازي. ولكني أبحث عن معلومات تدور حول شيءٍ ما، ولنقُل السيرة الذاتية لشخص بعينه. النص الناتج عن هذا البحث الذي أقرؤه على شاشة جهازي له معنًى بالنسبة إليَّ. وعند هذا المستوى، تصبح الأداة الحوسبية التي أتفاعل معها نظام معالجة معلومات دلالية.

بالطبع يمكن أن تكون هذه المعلومات أيَّ شيء عن البيئة المادية أو الاجتماعية أو الثقافية، أو عن الماضي، أو عن خواطر وأفكار أعلن عنها أصحابها، أو حتى عن الأفكار الداخلية لشخصٍ ما إذا حدث وسُجلت أو خُزنت في مكانٍ ما. وحسبما أشار عالِم الكمبيوتر بول روزنبلوم، فإن القاسم المشترك بين المعلومات الدلالية وغير الدلالية هو ضرورة التعبير عنها باستخدام وسيط مادي مثل الإشارات الكهربية أو الحالات المغناطيسية أو العلامات على الورق؛ وأن من شأنها كذلك أن تزيل اللبس.

هل المعلومات هي المعرفة؟

لنضرب المثل على المعلومات الدلالية بالسيرة الذاتية عن شخصٍ ما. بالاطلاع عليها، يمكنني بالتأكيد أن أزعم أني أملك «معرفة» عن ذلك الشخص. وهذا يشير إلى مصدر غموض آخر بشأن مفهوم المعلومات في اللغة العادية، ألا وهو الخلط بين المعلومات والمعرفة.

لم يكن لدى الشاعر تي إس إليوت شكٌّ في وجود فرقٍ بينهما. ففي مسرحيته «الصخرة» (١٩٣٤) طرح سؤالًا شهيرًا، وهو:

> أين الحكمة التي افتقدناها في المعرفة؟
> وأين المعرفة التي افتقدناها في المعلومات؟

لا يخفى أن إليوت كان يشير ضمنيًّا إلى تسلسل هرمي: أن الحكمة تعلو على المعرفة، والمعرفة تعلو على المعلومات.

يتحاشى علماء الكمبيوتر بوجه عام الحديثَ عن الحكمة؛ حيث إنها تتخطى نطاق اختصاصهم. لكنهم لم يستقروا بعدُ إلى حدٍّ ما بشأن الفرْق بين المعرفة والمعلومات، على الأقل في بعض السياقات. على سبيل المثال، في مجال الذكاء الاصطناعي المتفرع من علم الكمبيوتر، كان «تمثيل» المعرفة مشكلة أساسية وطويلة الأمد، وهي تدور حول طريقة تمثيل المعرفة عن العالم في ذاكرة الكمبيوتر. نوع آخر من المشاكل التي يدرسونها هو طريقة «استنباط الاستدلالات» من مجموعة معارف. إن أنواع الأشياء التي يعتبرها

باحثو الذكاء الاصطناعي معرفة تتضمن الحقائق (مثل «كل البشر فانون»)، والنظريات (مثل «التطور عن طريق الانتقاء الطبيعي»)، والقوانين (مثل «لكل فعل ردٌّ فعل مساوٍ له في المقدار ومضاد له في الاتجاه»)، والمعتقدات (مثل «للكون إله»)، والقواعد (مثل «يجب التوقُّف التام عند إشارة الوقوف»)، والإجراءات (مثل «طريقة طهي حساء المأكولات البحرية»)، وغير ذلك. ولكن الطريقة التي تشكِّل بها هذه الكياناتُ المعرفةَ دونًا عن المعلومات غير معروفة إلى حد كبير. كما قد يزعم الباحثون في الذكاء الاصطناعي أيضًا أنهم يعالجون المعرفة وليس المعلومات في فرع تخصصهم من علم الكمبيوتر؛ لكن يبدو أنهم ليس لديهم تفسير لسبب اهتمامهم بالمعرفة دونًا عن المعلومات.

في تخصصٍ آخر يسمَّى «التنقيب في البيانات»، ينصبُّ الاهتمام على «اكتشاف المعرفة» من كميات هائلة من البيانات. بعض الباحثين في مجال التنقيب في البيانات يعرِّفون المعرفة بأنها أنماط «مثيرة للاهتمام» و«مفيدة» أو أنماط منتظمة مخفية في قواعد البيانات الضخمة. ويفرِّقون بين اكتشاف المعرفة واسترداد المعلومات (وهذا نوع آخر من أنشطة الحوسبة)، حيث إن الأخير يهتم باستعادة المعلومات «المفيدة» من قواعد البيانات على أساس استعلامٍ ما، في حين يشير الأول إلى المعرفة التي هي أكثر من مجرد معلومات «مفيدة» أو مجرد أنماط منتظمة: حيث لا بد أن تكون هذه المعلومات «مثيرة للاهتمام» بشكلٍ ذي أهمية كبيرة كي ترتقي إلى رتبة المعرفة. وكمثل تي إس إليوت، يضع الباحثون في مجال التنقيب في البيانات المعرفةَ في مرتبةٍ أعلى من مرتبة المعلومات. وعلى أية حال، يرتبط التنقيب في البيانات بمعالجة المعرفة أكثر من ارتباطه باسترداد المعلومات.

طرح فيلسوف الحوسبة لوتشانو فلوريدي وجهةَ النظر التالية عن العلاقة بين المعلومات والمعرفة. يوجد «تشابه عائلي» بين المعلومات والمعرفة. فكلاهما كِيان ذو معنًى، ولكن يدور وجه الاختلاف حول أن عناصر المعلومات منفصلة مثل اللبنات، بينما المعرفة تربط عناصر المعلومات ببعضها بحيث يستطيع المرء استنباط استدلالات جديدة عن طريق تلك العلاقات.

لنضرب مثالًا: لنفترض أنني كنت أقود سيارتي وسمعت في مذياع السيارة أن علماء الفيزياء في جنيف قد اكتشفوا جسيمًا أساسيًّا يسمَّى بوزون هيجز. لا شك أن هذه الحقيقة الجديدة («جسيم بوزون هيجز موجود») تمثِّل معلومة جديدة بالنسبة إليَّ. إنني يمكنني حتى أن أعتقد أني اكتسبت معرفةً جديدة. لكن سيكون هذا الاعتقاد وهمًا ما لم أربط تلك المعلومة بعناصر معلومات أخرى مرتبطة بالجسيمات الأساسية وعلم الكون. وكذلك

لن أتمكَّن من قياس أهمية تلك المعلومة. إن علماء الفيزياء يحوزون شبكة متكاملة من الحقائق والنظريات والقوانين وغير ذلك عن الجسيمات دون الذرية وعن بنية الكون بما يمكِّنهم من استيعاب تلك الحقيقة الجديدة وفهْم أهميتها أو تبعاتها. إنهم يحوزون المعرفة اللازمة لذلك، أما أنا فلم أكتسب سوى معلومة جديدة.

هل المعلومات تُعَد «بيانات»؟

لَمَّا ذكرت «التنقيب في البيانات»، طرحت مصطلحًا جديدًا ذا صلة كبيرة وهو «البيانات». وهنا يكمُن مصدرٌ آخر من مصادر الغموض في فهْمنا لمفهوم المعلومات، لا سيما في أوساط علم الكمبيوتر.

أشار عالِم الكمبيوتر دونالد كنوث لهذا الإشكال — بل اختلاط المعاني — منذ عام ١٩٦٦، وهو الوقت الذي ظهر فيه علم الكمبيوتر باعتباره مجالًا علميًّا قائمًا بذاته، وكان يتطلب صياغة مفاهيم جديدة وتوضيح مفاهيم قديمة. ذكر كنوث أن في العلم ثمَّة ما يبدو أنه اختلاط من نوعٍ ما فيما يتعلَّق بمعاني المصطلحين «المعلومات» و«البيانات». فعندما يُجري عالمٌ ما تجربةً تحتوي على قياس، فإن ما يستنبطه قد يكون واحدًا من أربعة كِيانات وهي: القيم «الحقيقية» لما تم قياسه؛ والقيم التي حصل عليها بالفعل أو التقديرات التقريبية للقيم الصحيحة؛ وتمثيل تلك القيم؛ والمفاهيم التي يستخرجها العالِم من تحليل القياسات. وشدَّد كنوث أن كلمة «بيانات» تنطبق وتتلاءم أكثرَ مع الكيان الثالث من تلك الكيانات. وعلى حد قول كنوث باعتباره عالِم كمبيوتر، فإن البيانات هي «تمثيل» المعلومات المستخرجة عن طريق الملاحظة أو القياس بصورة دقيقة. ومن ثَم بناءً على رأيه، فإن المعلومات تسبق البيانات. ولكن عمليًّا، فالعلاقة بين المعلومات والبيانات غامضة مثلها مثل العلاقة بين المعلومات والمعرفة. وهنا، لا يسعني سوى الاستشهاد ببعض وجهات النظر المختلفة عن هذه العلاقة.

يرى عالِم الأنظمة والإدارة البارز راسل أكوف أن البيانات تُعَد نتيجة الملاحظات؛ أي أنها تمثيلات للأشياء والأحداث. أما بشأن المعلومات، فتخيَّل أكوف أن أحدًا يطرح بعض الأسئلة التي تنطوي على بيانات ثم تمر بـ «المعالجة» (على الأرجح من قِبل إنسان أو آلة) لتقديم إجابات عليها، وتلك الإجابات هي المعلومات. وبذلك يرى أكوف أن البيانات تسبق المعلومات، وهذا مناقضٌ لرأي كنوث.

ويرى لوتشانو فلوريدي أيضًا أن البيانات تسبق المعلومات ولكن بمعنًى مختلف. وفقًا لرأي فلوريدي، لا توجد البيانات إلا في غياب الاتساق بين حالتَي نظامٍ ما. فعلى حد تعبيره، توجد الوحدة الواحدة من البيانات (بالإنجليزية datum وهي الصيغة المفردة غير الشائعة الاستخدام لكلمة data) أينما وُجد المتغيران x وy بحيث x لا يساوي y. هكذا طبقًا لفلوريدي فإن البيانات عبارة عن حالة ليس لها معنًى في حد ذاتها باستثناء أنها تدُل على وجود فرْق. فعندما أقترب من إشارة مرور، على سبيل المثال، رؤيتي لإشارة حمراء عبارة عن وحدة واحدة من البيانات لأنها كانت من الممكن أن تكون لونًا آخر؛ الأصفر أو الأخضر.

بناءً على هذا التعريف للبيانات، يعرِّف فلوريدي المعلومات بأنها عنصر أو عدة عناصر من البيانات المبنية وفقًا لقواعدَ معينة، والتي تكون ذات معنًى. وفي اصطلاح اللغويين، تكون المعلومات بيانات عندما تمتلك كلًّا من البناء والمعنى الدلالي. وبذلك تصبح رؤيتي للإشارة الحمراء — وحدة واحدة من البيانات — معلوماتٍ لأن معنى الإشارة الحمراء هي «ضرورة توقُّف سائقي المركبات عند إشارة المرور». وإذا لم أربط هذا الإجراء بالإشارة الحمراء، فستظل الإشارة الحمراء مجرد وحدة بيانات.

لنضرب مثالًا أخيرًا، يرى الباحثان في الذكاء الاصطناعي جيفري شراجر وبات لانجلي أن البيانات لا تنتج عن الملاحظة، بل إن الملاحظة هي البيانات؛ بمعنًى أدق، ما يلاحَظ يسجَّل بشكل انتقائي كي يرتقي إلى مرتبة البيانات. ولا تظهر المعلومات في مخططهم للأشياء.

وجهة نظر المبرمجين

هذه الأمثلة كافية لتوضيح الغموض في وجه الصلة بين المعلومات والبيانات من وجهات نظر مختلفة. لكن لنرجع إلى كنوث. أعتقد أن تعريفه يبرز إلى حدٍّ كبير وجهةَ نظر علماء الكمبيوتر المتخصصين في ضربٍ آخرَ من علم الكمبيوتر اسمه برمجة الكمبيوتر — وهي الأساليب التي يستخدمها الإنسان كي يملي مهمةً حوسبية على الكمبيوتر (وسأناقش هذا الموضوع لاحقًا في هذا الكتاب). وعلى الرغم من أن المبرمجين ومنظِّري البرمجة يؤيدون بألسنتهم فكرة أن الحوسبة عبارة عن معالجة للمعلومات، فإنهم لا يهتمون بـ «المعلومات» بوجه عام، بل إنهم يهتمون أكثرَ بفكرة كنوث عن البيانات. بعبارة أدق، يهتم أولئك بالبيانات باعتبارها العناصر الأساسية (عناصر البيانات) التي تُنفذ العمليات الحوسبية

عليها؛ ومن ثَم فهم منشغلون بتصنيف عناصر البيانات (أنواع البيانات)، وقواعد تمثيل عناصر البيانات المعقَّدة (هياكل البيانات)، وقواعد استخدام عناصر البيانات ومعالجتها وتطويرها بهدف إنتاج عناصر بيانات جديدة. بالنسبة إلى علماء الكمبيوتر هؤلاء، البيانات هي ما تهمهم، وليس المعلومات أو المعرفة. لمزيد من الدقة، يسلِّم المبرمجون جدلًا بأن المعلومات «موجودة» في «العالم الحقيقي». لكن المسألة التي تثير اهتمامهم هي طريقة تمثيل معلومات العالم الحقيقي في شكلٍ لا يتلاءم مع الحوسبة التلقائية فحسب، بل مع فهْم الإنسان أيضًا. (غني عن التوضيح أن الممارسين الآخرين مثل علماء التاريخ والإحصاء والعلماء التجريبيين لا ينظرون عادةً إلى البيانات من تلك الزاوية.)

سأتناول هذا الموضوع بمزيدٍ من التفصيل في موضع لاحق من الكتاب. لكن لنضرب مثالًا بسيطًا للغاية على رؤية المبرمجين للبيانات: في البيئة الجامعية، ستكون ثمَّة معلومات في مكتب أمين السجلات عن الطلاب المسجلين، ومنها أسماء الطلاب وتواريخ ميلادهم وعناوين منازلهم وعناوين البريد الإلكتروني، وأسماء الوالدين أو أولياء الأمور، وموضوعات التخصص والدورات التدريبية التي درسوها والدرجات التي حصلوا عليها والمنح الدراسية والرسوم المدفوعة، وغير ذلك. تحتاج إدارة الجامعة إلى نظام ينظِّم هذه المعلومات بطريقة منهجية (قاعدة بيانات) بحيث يمكن استرداد المعلومات التي تخص أيَّ طالب بعينه بدقة وبسرعة، وكذا إدخال معلومات جديدة عن الطلاب الحاليِّين أو الجدد، وتتبُّع تقدُّم فرادى الطلاب تتبُّعًا فعالًا، وجمع الإحصاءات عن الطلاب جملةً أو عن مجموعة منهم. المبرمج الذي توكل إليه هذه المهمة لا تعنيه المعلومات في حد ذاتها، بل تعنيه — بناءً على طبيعة المعلومات — طريقةَ تحديد عناصر البيانات الأساسية التي تمثِّل معلومات الطالب، وشكل هياكل البيانات التي تمثِّل عناصر البيانات، وبناء قاعدة بيانات تسهِّل على إدارة الجامعة المهامَّ الحوسبية التي تريدها.

البنيات الرمزية باعتبارها القاسم المشترك

بدأت هذا الفصل باقتراح أن المقوم الأساسي للحوسبة هو المعلومات، وأن الكمبيوتر عبارة عن أوتوماتون يعالج المعلومات، وعليه يكون علم الكمبيوتر هو دراسة معالجة المعلومات.

ولكننا رأينا أيضًا أن بعض علماء الكمبيوتر (مثل الباحثين في الذكاء الاصطناعي) يرون أن المقوم الأساسي للحوسبة هو المعرفة وليس المعلومات؛ ويرى آخرون (مثل المبرمجين ومنظِّري البرمجة) أن البيانات هي المقوم الأساسي وليس المعلومات. سنتعرَّف

على الاستخدامات المتنوعة لهذه الكيانات الثلاثة من النموذج التالي للمصطلحات الواردة في المؤلفات عن الحوسبة (والتي قد ورد بعضها بالفعل في هذا الفصل، وسيظهر غيرها في فصول لاحقة):

> نوع البيانات، عنصر البيانات، هيكل البيانات، قاعدة البيانات، معالجة البيانات، التنقيب في البيانات، البيانات الضخمة …
>
> معالجة المعلومات، نظام المعلومات، علم المعلومات، هيكل المعلومات، تنظيم المعلومات، تكنولوجيا المعلومات، تخزين المعلومات واستردادها، نظرية المعلومات …
>
> قاعدة المعرفة، نظام المعرفة، تمثيل المعرفة، هيكل المعرفة، نظرية المعرفة، المعرفة التقريرية، المعرفة الإجرائية، اكتشاف المعرفة، هندسة المعرفة، مستوى المعرفة …

هل يمكننا إذن اختزال هذه الكيانات الثلاثة — المعلومات والبيانات والمعرفة — إلى قاسم مشترك؟ في الحقيقة يمكننا ذلك. عالم الكمبيوتر بول روزنبلوم جعل المعلومات نظيرًا «للرموز»، ولكن يمكننا المضي إلى ما هو أبعد من ذلك. فبقدرِ ما يتعلق الأمر بعلم الكمبيوتر، يمكن التعبير عن هذه الكيانات الثلاثة (وعادةً ما يعبَّر عنها بالفعل) باستخدام الرموز — أو بالأحرى باستخدام أنظمة الرموز أو «البنيات الرمزية» — أي، الكيانات التي «ترمز إلى» كيانات أخرى أو تمثلها أو تدُل عليها.

تحتاج الرموز إلى وسيط للتعبير عنها، مثل العلامات على الورقة. على سبيل المثال، النص «جسيم بوزون هيجز موجود» عبارة عن بنية رمزية تتكون مكوناتها الرمزية من حروف أبجدية تشير إلى وحدات صوتية أو صُويتات بالإضافة إلى رمز «المسافة»، وعندما تُجمع هذه الرموز بعضها مع بعض فإنها تمثِّل شيئًا عن العالم المادي. في نظر غير المتخصص في الفيزياء، فهذا النص عبارة عن معلومة، ولكن في نظر عالِم فيزياء الجسيمات، يصبح هذا النص جزءًا أساسيًّا من نظام معرفته فيما يخص الجسيمات الأساسية. وعلى الرغم من ذلك، فإن معرفة عالِم الفيزياء التي تتيح له فهْم هذه المعلومة هي في حد ذاتها عبارة عن بنية رمزية بالغة التعقيد مخزَّنة في دماغه أو مكتوبة ضمن نص في الكتب أو المقالات. وفكرة كنوث عن البيانات بأنها تمثيل للمعلومات تعني أن تلك البيانات أيضًا عبارة عن بنيات رمزية تمثل بنيات رمزية أخرى تدل على معلومات. حتى المعلومات «غير الدلالية» في نظرية المعلومات — وحدات البت والبايت — تُمثَّل بالرموز

الفيزيائية داخل الكمبيوتر، مثل مستويات الجهد أو الحالات المغناطيسية، أو تُمثَّل على الورق باستخدام سلاسل مكوَّنة من 0 و1.

ومن ثَم من حيث الأساس الأكثر جوهرية، فإن مقوم الحوسبة هو البنيات الرمزية. إذن، الحوسبة هي معالجة الرموز. وأي أوتوماتون قادر على معالجة بنيات رمزية هو كمبيوتر. وعلى حد تعبير بيرليس ونويل وسايمون، يمكن اختزال «الظواهر» المرتبطة بأجهزة الكمبيوتر في نهاية الأمر إلى بنيات رمزية وعمليات معالجتها. وبذلك يصبح علم الكمبيوتر في نهاية المطاف هو علم المعالجة التلقائية للرموز، وهي الرؤية التي أكَّد ألين نويل وهيربرت سايمون عليها. يجوز لنا أن نطلق على البنيات الرمزية اسم المعلومات أو البيانات أو المعرفة بناءً على «الثقافة» الخاصة التي ننتمي إليها ضمن مجال علم الكمبيوتر.

وهذا المفهوم — الذي يرى أن الحوسبة هي في الأخير معالجة الرموز، وأن الكمبيوتر هو أوتوماتون يعالج الرموز، وأن علم الكمبيوتر هو علم معالجة الرموز — هو ما يميِّز علم الكمبيوتر عن باقي العلوم. أما فيما يتعلق بغرابته، فهذا ما سنتناوله في فصل لاحق.

الفصل الثاني

الأدوات الحوسبية

إننا ننظر إلى الكمبيوتر على أنه محور الحوسبة؛ ومن ثَم، علم الكمبيوتر أيضًا. وهذا صحيح. لكن ينبغي الانتباه إلى بعض التحفظات.

أولًا: ربما تختلف الآراء بشأنِ ما يشكِّله «الكمبيوتر» بالتحديد. يرى البعض جهاز الكمبيوتر الشيءَ المادي الذي يعملون عليه يوميًّا (جهاز الكمبيوتر المحمول أو الكمبيوتر المكتبي في العمل). يراه آخرون نظامًا كاملًا تحت تصرُّفهم، بما في ذلك أدوات تيسير المهام مثل خدمة البريد الإلكتروني ومعالجة الكلمات والوصول إلى قواعد البيانات، وغير ذلك. لكن لا يزال البعض يربطه بنموذج رياضي بالكامل يسمَّى آلة تورنج (التي سنتناولها لاحقًا في هذا الفصل).

ثانيًا: بناءً على قبول أن الكمبيوتر هو أوتوماتون لمعالجة الرموز، هناك أيضًا أدوات أخرى لمعالجة الرموز مرتبطة بالكمبيوتر، ولكن يبدو أنها تختلف قليلًا مع فكرتنا البديهية عن «الكمبيوتر». ومن ثَم، ينبغي لنا أن نتحلى بمزيد من الانتقائية في نظرتنا إلى الأدوات التي تشارك في عملية الحوسبة؛ ومن هنا جاء مصطلح «الأدوات الحوسبية». في هذا الفصل، سنتناول طبيعةَ الأدوات الحوسبية.

في الفصل الأول، بدا الكمبيوتر (بشكل أو بآخر) وكأنه صندوق أسود. جلُّ ما ذكرناه أنه أوتوماتون لمعالجة الرموز: فهو جهاز يقبل البنيات الرمزية (التي تشير إلى المعلومات أو البيانات أو المعرفة حسبما يقتضي الحال) في صورة مدخلات وينتج (بفعله الخاص) بنيات رمزية في صورة مخرجات.

عندما ننوي فتْح هذا الصندوق الأسود، نجد أنه يشبه إلى حدٍّ ما مجموعةً من الصناديق المتداخلة: بمعنى أنه بالداخل يوجد صندوق أو عدة صناديق أصغر؛ كلما فتحنا صندوقًا من الصناديق الداخلية، تبيَّن لنا أنه توجد صناديق أصغر متشعِّبة منه.

وهكذا دواليك. بالطبع درجة تداخل الصناديق السوداء لها نهاية، فعاجلًا أو آجلًا سنصل إلى الصناديق الأولية.

يُظهر كلٌّ من العالَمين الطبيعي والاصطناعي أمثلةً على هذه الظاهرة التي تسمَّى «التسلسل الهرمي». العديد من الأنظمة الفيزيائية والأحيائية والاجتماعية والتكنولوجية لها هيكل ذو تسلسل هرمي. الفرْق بين التسلسلات الهرمية الطبيعية (مثل الأنظمة الحية) والاصطناعية (مثل الأنظمة الثقافية أو التكنولوجية) هو أن العلماء عليهم «اكتشاف» التسلسلات الطبيعية و«ابتكار» التسلسلات الاصطناعية.

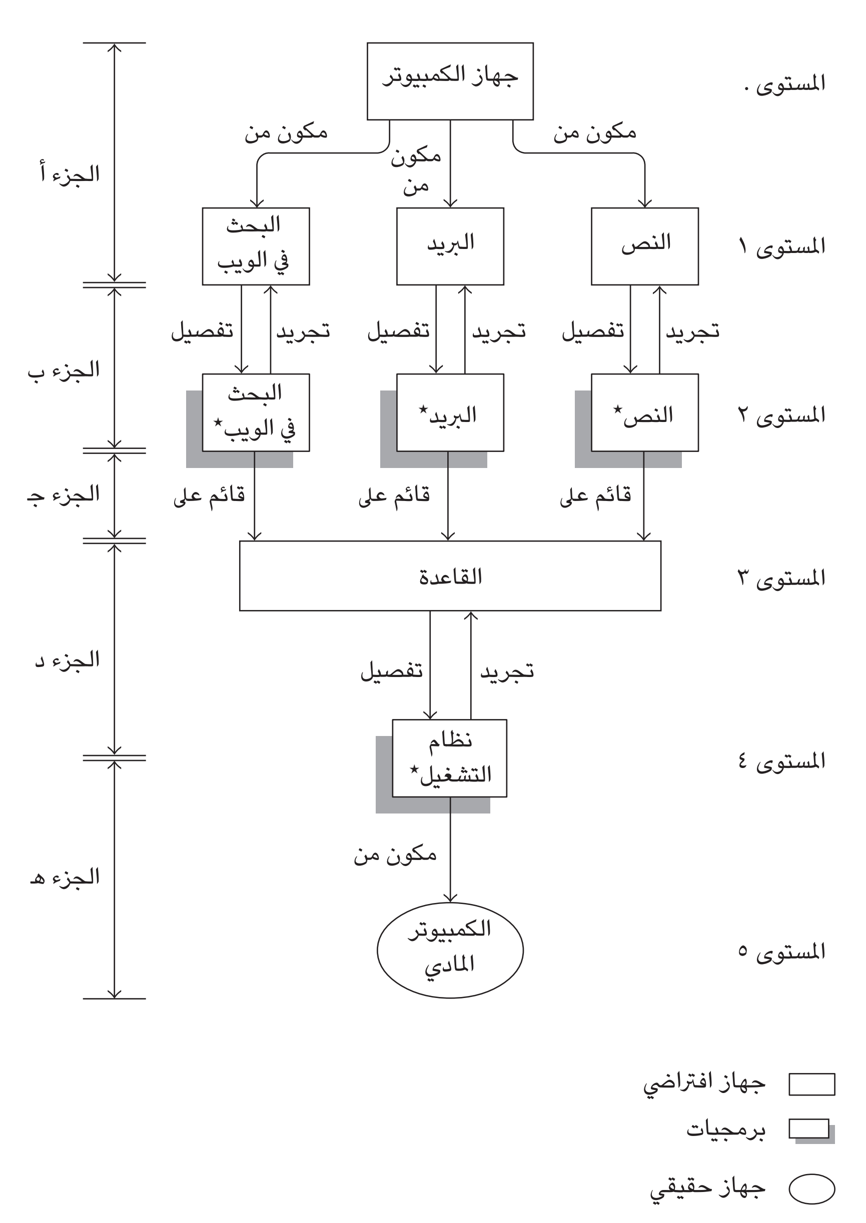
الكمبيوتر الحديث عبارة عن نظام من الأدوات الحاسوبية ذي تسلسل هرمي. وبذلك يصبح ابتكار قواعد ومبادئ التسلسل الهرمي وفهْمها وتطبيقها مجالًا فرعيًّا من علم الكمبيوتر.

هناك سبب وراء وجود التسلسلات الهرمية في كلٍّ من العالمين الطبيعي والاصطناعي، وإننا مدينون بتلك الرؤية — على وجه الخصوص — إلى العالِم الموسوعي هيربرت سايمون. قال سايمون إن التنظيم الهرمي وسيلة لإدارة «وجه التعقيد» في كيانٍ ما. وفي رأي سايمون، يصبح الكيان معقدًا إذا كان يتكون من عدة مكونات تتفاعل بعضها مع بعض بطريقة غير بسيطة (أو غير واضحة). وكما سنرى، يُظهر الكمبيوتر هذا النوع من التعقيد، ومن ثَم فهو أيضًا يتألف من نظام هرمي. إن مصمِّمي أنظمة أجهزة الكمبيوتر ومنفذيها مجبرون على هيكلتها بناءً على مبادئ التسلسل الهرمي وقواعده. ويتولى علماء الكمبيوتر مسئولية ابتكار هذه القواعد والمبادئ.

التسلسل الهرمي التركيبي

بوجه عام، تتكوَّن الأنظمة الهرمية من مكونات مقسمة إلى مستويين أو أكثر. وتُعنى أشهر مبادئ التسلسل الهرمي بالعلاقة بين المكونات داخل المستويات وفيما بينها.

يصوِّر الشكل ٢-١ ما أسميه «جهاز الكمبيوتر». (ماديًّا، قد يكون كمبيوترًا مكتبيًّا أو محمولًا أو جهازًا لوحيًّا أو حتى هاتفًا ذكيًّا. ولغرض التسهيل، سأفترض أنه واحد من أول نوعين.) لنفترض أنني لا أستخدم هذا الجهاز إلا في ثلاثة أنواع من المهام وهي: كتابة النصوص (كما أفعل الآن)، وإرسال رسائل البريد الإلكتروني، والبحث في شبكة الويب عبْر الإنترنت. وبذلك، أرى أن الكمبيوتر يتكون من ثلاث «أدوات حوسبية» سأسميها «النص» و«البريد الإلكتروني» و«البحث في الويب» (المستوى الأول من الشكل ٢-١)، على التوالي.



شكل ٢-١: التجريد والتسلسل الهرمي داخل نظام الكمبيوتر.

كل أداة عبارةٌ عن أداة حوسبية لمعالجة الرموز. وكل أداة تتميَّز (من وجهة نظري بصفتي مستخدم الأداة) على أساس «إمكانيات» معينة. على سبيل المثال، توفِّر أداة «النص» واجهةَ المستخدم التي تتيح لي إدخال سلسلة من الحروف، وإعطاء الأوامر لمحاذاة الهوامش، وتعيين المسافات بين السطور، وتقسيم الصفحات، وبدء فقرة جديدة، وتعيين المسافة البادئة، وإدخال رموز خاصة، وإضافة حواشٍ سفلية وتعليقات ختامية، وتغيير الخط إلى مائل أو سميك، وغير ذلك. كما تتيح لي أيضًا إدخال سلسلة من الحروف التي تدخل في النص — باستخدام الأوامر — بحيث يمكنني استخدامه أو استرداده في وقت لاحق.

من وجهة نظري، أداة «النص» هي «جهاز الكمبيوتر» عندما أكتب مقالًا أو كتابًا (كما أفعل الآن)، وكذلك الحال مع أداة «البريد الإلكتروني» عندما أرسل رسائل البريد الإلكتروني وأداة «البحث في الويب» عندما أبحث في شبكة الويب. بعبارةٍ أدق، يتوافر لديَّ ثلاث صور خيالية مختلفة بديلة لماهية «جهاز الكمبيوتر». يشير علماء الكمبيوتر إلى تلك الأدوات الخيالية باسم «الأجهزة الافتراضية»، وإنشاء هذه الأجهزة الافتراضية وتحليلها وفهمها من أهم شواغل علم الكمبيوتر. فهي تشكِّل إحدى الظواهر التي تحيط بأجهزة الكمبيوتر التي أشار إليها بيرليس ونيويل وسايمون.

ويستخدم علماء الكمبيوتر مصطلح «المعمارية» بوجه عام للإشارة إلى البناء المنطقي أو الوظيفي للأدوات الحوسبية. (المصطلح «معمارية الكمبيوتر» له معنًى متخصص أكثر سنتناوله لاحقًا.) من وجهة نظري (أو وجهة نظر أي مستخدم آخر)، أداة «النص» الحوسبية لها معمارية خاصة أراها كما يلي: تتكوَّن من مترجم يترجم الأوامر وينفِّذها، وذاكرة مؤقتة أو عاملة محتواها هو النص الذي أكتبه، وذاكرة دائمة أو طويلة الأجل تحتفظ بكل النصوص المختلفة في صورة ملفات اخترت أن أحفظها، وقنوات مدخلات تنقل تدفقات الحروف والأوامر إلى الجهاز، وقنوات مخرجات تتيح عرض النصوص على الشاشة أو تتيح طباعتها (كنسخة ورقية). هذه المكونات «وظيفية»: ربما لا أعرف (أو لا أهتم على وجه الخصوص) بالوسائط الفعلية التي تتوافر فيها هذه المكونات. ولأنها تميز كلَّ ما أحتاج أنا (بصفتي مستخدمًا) إلى معرفته عن أداة «النص» حتى أتمكَّن من استخدامها، سنطلق عليها اسم معمارية أداة «النص».

وبالمثل، عند إرسال رسائل البريد الإلكتروني، تصبح أداة «البريد الإلكتروني» هي «جهاز الكمبيوتر»؛ أي إنها أداة حوسبية افتراضية. وهي أيضًا معالج للرموز. فهي تظهر واجهة مستخدم تمكنني من تحديد مستلم أو أكثر لرسالتي، وربط بنية رمزية أخرى أو

أكثر (في شكل نصوص أو صور) في صورة مرفقات تُرسل مع الرسالة، وكتابة الرسالة، وإرسالها إلى المستلم أو المستلمين. وتتشابه معماريتها مع معمارية أداة «النص»؛ حيث إنها تُظهر أنواع المكونات ذاتها. فهذه الأداة بإمكانها ترجمة الأوامر، ولها قنوات مدخلات بما يمكِّن تدفُّق الحروف من التجمُّع في الذاكرة العاملة، وذاكرة طويلة الأجل لحفظ رسائلي بقدرِ ما أرغب من وقت، وقنوات مخرجات لعرض محتويات رسالة البريد الإلكتروني على الشاشة أو لطباعتها. بالإضافة إلى ذلك، يمكن لأداة «البريد الإلكتروني» الوصول إلى أنواع أخرى من الذاكرة الطويلة الأجل التي تحفظ البنيات الرمزية (كالنصوص والصور) التي يمكن إرفاقها مع الرسالة؛ لكن واحدة من الذاكرات الطويلة الأجل «خاصة» بالنسبة لـ «جهاز الكمبيوتر»، ومن ثَم لا يمكن لأحد غيري الوصول إليها، في حين تكون الأخرى «عامة»؛ أي إنها مشتركة مع مستخدمي أجهزة الكمبيوتر الأخرى.

وأخيرًا، توجد أداة «البحث في الويب». معمارية هذه الأداة مشابهة للأداتين السابقتين وهي: مترجم الأوامر، وذاكرة مشتركة/عامة (شبكة الويب) يمكن الوصول إلى محتوياتها (صفحات الويب)، وذاكرة عاملة خاصة تحتفظ (مؤقتًا) بالمحتويات التي يتم الوصول إليها من الذاكرة المشتركة، وذاكرة خاصة طويلة الأجل تحفظ هذه المحتويات، وقنوات مدخلات ومخرجات.

يتكوَّن التسلسل الهرمي الموضَّح في الجزء أ من الشكل ٢-١ من مستويين. في المستوى الأعلى (صفر) توجد أداة حوسبية واحدة وهي «جهاز الكمبيوتر»، ولكن المستوى الأدنى منه (الأول) يوضح أن «جهاز الكمبيوتر» يتكون من ثلاث أدوات مستقلة. هذا المستوى يُعَد صندوق أدواتي، إذا جاز التعبير. إن هذا النوع من التسلسل الهرمي، عندما يتكون الكيان A من الكيانات α، وβ، وγ، وغيرها، شائع في الأنظمة المعقَّدة من أي نوع، سواء كانت طبيعية أو اصطناعية. لا شك أن هذا التسلسل الهرمي يميِّز الأدوات الحوسبية. ولا يوجد مصطلح مقبول بشكل عام له، لذلك سنسميه «التسلسل الهرمي التركيبي».

التجريد/التفصيل

كما أشرنا، يتكون المستوى الأول في الشكل ٢-١ من ثلاث أدوات حوسبية ذات معمارية متشابهة. فتتكون كلُّ أداة من ذاكرة مشتركة وذاكرة خاصة طويلة الأجل، وذاكرة عاملة خاصة، وقناة أو أكثر للمدخلات والمخرجات، ومترجم أو أكثر للأوامر.

لكن لا بد أن الأدوات الحوسبية الثلاث هذه قد مرَّت بـ «مرحلة تنفيذ» كي تكون أدوات عاملة فعلية: على سبيل المثال، لا بد أن شخصًا صمَّم أداة حوسبية ونفذها كي

«تؤدي دور» أداة «النص» عند تنشيطها، مع إخفاء تفاصيل الآليات التي تحققت بها أداة «النص». لنُشِر إلى هذه الأداة المنفَّذة باسم «النص*» (المستوى الثاني في الشكل ٢-١)؛ وهي عبارة عن برنامج كمبيوتر، أو أحد البرمجيات. العلاقة بين الأداتين «النص» و«النص*» هي التجريد/التفصيل (الجزء ب في الشكل ٢-١):

> «تجريد» الكيان E هو الكيان الآخر e الذي لا يكشف إلا عن سمات الكيان E التي تعتبر ذات صلة في أحد السياقات، ولكنه يخفي سماتٍ أخرى تعتبر غير ذات صلة (في ذلك السياق). وعلى العكس، «تفصيل» الكيان e هو الكيان الآخر E الذي يكشف عن السمات التي كانت غائبة أو مخفية في الكيان e.

أداة «النص» تجريد لأداة «النص*»؛ وفي المقابل، أداة «النص*» تفصيل لأداة «النص». لاحظ أن التجريد/التفصيل أيضًا هو أحد مبادئ التسلسل الهرمي؛ حيث يأتي التجريد في مستوًى أعلى، ويأتي التفصيل في مستوًى أدنى. لاحظ أيضًا أن عمليات التجريد والتفصيل تعتمد على السياق. إذ يمكن تجريد الكيان E بطرقٍ مختلفة لإنتاج كيانين أو عدة كيانات ذات مستوًى أعلى $e1$، و$e2$، و…، وeN. وفي المقابل، يمكن تفصيل الكيان e ذاته بطريقتين مختلفتين إضافيتين لإنتاج كيانات مختلفة ذات مستوًى أدنى $E1$، و$E2$، و…، وEn.

ولاتخاذ مبدأ التجريد/التفصيل باعتباره طريقةً لإدارة أوجه تعقيد الأدوات الحوسبية تاريخ زاخر يعود إلى الأعوام الأولى لميلاد الحوسبة. ربما كان الشخص الذي أسهم في زيادة وعي مجتمع علم الكمبيوتر الناشئ في ستينيات القرن العشرين بشأن أهمية التسلسل الهرمي هو إدسخر ديكسترا. سنرى لاحقًا أهميته الخاصة في عملية «بناء» الأدوات الحوسبية، ولكن في الوقت الحالي بحسب القارئ أن يقدِّر كيف يمكن أن تُفهم الأداة الحوسبية المعقَّدة فيما يتعلق بمبدأ التجريد/التفصيل، مثلما يمكن فهْم الأداة الحوسبية المعقَّدة فيما يتعلق بالتسلسل الهرمي التركيبي.

التسلسل الهرمي بحكم البناء

سنتوقف الآن عن النظر إلى «جهاز الكمبيوتر» من منظور المستخدم. إننا الآن نقف في موقف الذين أنشئوا «جهاز الكمبيوتر» بالفعل: مُنشئي الأداة. وبالمناسبة، ليس هؤلاء بجماعة متجانسة.

الأدوات «النص*» و«البريد الإلكتروني*» و«البحث في الويب*» هي على وجه الخصوص عبارة عن برامج كمبيوتر — برمجيات — أنشأها المبرمجون (مطورو البرمجيات كما يحبون أن يسموا أنفسهم اليوم) بناءً على بنية أساسية أسميها هنا «القاعدة» (الجزء جـ والمستوى الثالث في الشكل ٢-١). تتكون هذه البنية الأساسية أيضًا من مجموعة أدوات حوسبية يمكن أن يستخدمها مصمِّمو أداة «النص*» وغيرهم. إذن يصبح لدينا هنا نوع ثالث من التسلسل الهرمي وهو: التسلسل الهرمي بالبناء.

منذ الأيام الأولى لاختراع الكمبيوتر الرقمي، حرص المصمِّمون والباحثون على حماية المستخدم، بأقصى قدرٍ ممكن، من الحقائق المزعجة — والبغيضة في بعض الأحيان — الخاصة بالكمبيوتر المادي لإضفاء المزيد من السهولة على حياة المستخدم. ما فتئوا يطمحون إلى تصميم واجهة مستخدم سلسة وبسيطة وقريبة إلى اهتمامات المستخدم الخاصة، وتبقى ضمن نطاق راحته. فالمهندس المدني أو مهندس الميكانيكا يريد تنفيذ عمليات حوسبية تأمر الكمبيوتر أن يحل معادلات الميكانيكا الهندسية، ويريد الروائي من الكمبيوتر أن يضطلع بدور أداة الكتابة، كما يريد المحاسب «أن ينفُض عن كاهله» بعضَ العمليات الحسابية المضجرة ويتركها للكمبيوتر، وهلم جرًّا. في كل حالة، يريد المستخدم المعنيُّ صورةً خيالية بأن الكمبيوتر قد صُمِّم خصوصًا لتلبية احتياجاته. بعد ذلك على مدار السنين نشب جدالٌ كبير حول ما إذا كان الأفضل دمج تلك البنى الأساسية والأدوات الخاصة بالمستخدم في الجهاز المادي (تثبيتها في الجهاز) أم توفيرها بطريقةٍ أكثر مرونة باستخدام البرمجيات. وبوجه عام، تم ترشيد تقسيم البنى الأساسية والأدوات عبْر هذا الانقسام بحسب الاحتياجات الخاصة لفئات المستخدمين الذين تُطوَّر أجهزة الكمبيوتر من أجلهم.

وكما أبرزنا، يقدِّم «جهاز الكمبيوتر» هذه الصورَ الخيالية للمستخدم الذي تنحصر اهتماماته في كتابة النصوص وإرسال رسائل البريد الإلكتروني واستقبالها والبحث في شبكة الويب. فيوفِّر «جهاز الكمبيوتر» البنية الأساسية للمستخدم لكتابة النصوص وكتابة رسائل البريد الإلكتروني وإرسالها والبحث عن المعلومات على شبكة الويب، تمامًا كما توفِّر «القاعدة» (في مستوًى أدنى) هذه البنية الأساسية من أجل بناء البرامج التي تكون بمثابة مجموعة أدوات للمستخدم.

لكن حتى مطوِّرو البرمجيات الذين أنشئوا هذه التجريدات بتنفيذ برنامج «النص*» وغيره من البرامج لا بد أن لديهم صورهم الخيالية؛ فهم أيضًا مستخدمون للكمبيوتر مع

أن تفاعلهم مع الكمبيوتر أكثر تعقيدًا من تفاعلي حينما أستخدم أداة «النص» أو «البريد الإلكتروني». يمكننا أن نسميهم «مبرمجو التطبيقات» أو «مطورو برمجيات التطبيقات»، ويجب حمايتهم هم أيضًا من بعض الحقائق بشأن الكمبيوتر المادي. فهم كذلك بحاجة إلى بنية أساسية يمكنهم العمل «معها» ويمكنهم إنشاء أجهزتهم الافتراضية بناءً «عليها».

في الشكل ٢-١، يمثِّل الكيان المسمَّى «القاعدة» هذا الأساس. وفي الحقيقة، هو تجريد لمجموعة من البرامج (نظام برمجيات) موضَّحة في الشكل باسم «نظام التشغيل*» (المستوى الرابع) والتي تنتمي إلى فئة من الأدوات الحوسبية التي يطلَق عليها «أنظمة التشغيل».

نظام التشغيل هو أداة التنسيق الرائعة؛ هو أداة الحماية الفعَّالة، إنه الساحر الإلكتروني الكبير. في بدايات تطوير نظام التشغيل في ستينيات القرن العشرين، كان يسمَّى «المشرف» أو «المسئول التنفيذي»، وهذان المصطلحان يبينان مسئولياته جيدًا. فوظيفته إدارة موارد الكمبيوتر المادي وتقديم مجموعة موحَّدة من الخدمات لكل المستخدمين، سواء كانوا مستخدمين عاديين أو مطوري برمجيات. تتضمَّن هذه الخدمات «أدوات التحميل» التي تقبل البرامج التي سيجري تنفيذها وتخصصها للمواقع المناسبة في الذاكرة؛ وإدارة الذاكرة (والتي تضمن عدم تعدي أو تداخل أحد برامج المستخدم مع الذاكرة التي يستخدمها برنامج آخر)؛ وتوفير ذاكرة افتراضية (والتي توهم المستخدمين بأن الذاكرة غير محدودة)؛ والتحكم في الأجهزة المادية (مثل الأقراص والطابعات والشاشات) التي تؤدي الوظائف الخاصة بالمدخلات والمخرجات؛ وتنظيم سعة تخزين المعلومات (أو البيانات أو المعرفة) في الذاكرات الطويلة الأجل بحيث يمكن الوصول إليها بسهولة وسرعة؛ وتنفيذ الإجراءات وفقًا لقواعد موحَّدة (تسمَّى «البروتوكولات») تمكِّن أحد البرامج على أحد أجهزة الكمبيوتر من طلب خدمة من برنامج على كمبيوتر آخر عبْر شبكة بينهما؛ وحماية برنامج أحد المستخدمين من أن يتلفه برنامج مستخدم آخر، سواء عن طريق الخطأ أو عن طريق أذًى متعمَّد من المستخدِم الآخر. البنية الأساسية التي تسمَّى «القاعدة» في الشكل ٢-١ توفِّر هذه الخدمات — وهي مجموعة من الأدوات الحوسبية؛ إنها تجريد لنظام التشغيل «نظام التشغيل*».

لكن نظام التشغيل ليس جدار حماية بالمعنى الحرفي بحيث يمنع كل التفاعل بين البرنامج المصمَّم في المستوى الأعلى منه (مثل «البريد الإلكتروني*»)، والكمبيوتر المادي في المستوى الأدنى منه. ففي نهاية الأمر، يعمل البرنامج بإصدار التعليمات أو الأوامر إلى

الكمبيوتر المادي، ومعظم هذه التعليمات سيترجمها الكمبيوتر المادي مباشرةً (وفي هذه الحالة، تسمَّى هذه التعليمات «تعليمات الآلة»). ما يفعله نظام التشغيل هو «تمرير» تعليمات الآلة إلى أجهزة الكمبيوتر المادية بطريقة محكمة، ويترجم التعليمات الأخرى بنفسه (مثل التعليمات الخاصة بمهام المدخلات والمخرجات).

يكاد هذا يصل بنا إلى نهاية التسلسل الهرمي المصوَّر في الشكل ٢-١. يظهر برنامج «نظام التشغيل*» على أنه مبني في مستوًى أعلى من الكمبيوتر المادي (المستوى الخامس). وفي الوقت الحالي، سنفترض أن الكمبيوتر المادي (يشيع استخدام لفظة «العتاد» عنه ببساطة) هو «الأداة الحقيقية» (أخيرًا)؛ بمعنى أنه لا يوجد شيء افتراضي فيه. سنرى أن هذا أيضًا عبارة عن صورة خيالية؛ حيث إن الكمبيوتر المادي له تسلسله الهرمي الداخلي وله مستوياته الخاصة من التجريد والتركيب والبناء. ولكن يمكننا على الأقل إنهاء المناقشة الحالية بالملاحظة التالية: يوفِّر الكمبيوتر المادي بنيةً أساسية وصندوقَ أدوات يحتوي على مجموعة كاملة من التعليمات (تعليمات الآلة)، ومجموعةً كاملة من أنواع البيانات (ارجع إلى الفصل الأول)، وأنماطًا لتنظيم التعليمات والبيانات في الذاكرة والوصول إليها، ومجموعةً معينة من الأدوات الأساسية الأخرى التي تتيح تشغيل البرامج (لا سيما نظام التشغيل) التي يمكن تنفيذها باستخدام الكمبيوتر المادي.

فئات الأدوات الحوسبية الثلاث

في كتاب حديث يتناول تاريخ ميلاد علم الكمبيوتر، علقتُ قائلًا إن السمة المميزة في علم الكمبيوتر تكمُن في فئاته الثلاث من الأدوات الحوسبية.

الفئة الأولى «مادية». هذه الأدوات — شأنها شأن كل الأشياء المادية التي ظهرت على مر التاريخ — تتبع قوانين الفيزياء في الطبيعة (مثل قانون أوم وقوانين الديناميكا الحرارية وقوانين الحركة لنيوتن وغيرها). تستهلك هذه الأدوات الطاقة وتولِّد الحرارة، وتنطوي على حركة فيزيائية (في بعض الحالات)، وتتحلل فيزيائيًّا وكيميائيًّا بمرور الوقت، وتشغل حيزًا فيزيائيًّا، وتستهلك وقتًا فيزيائيًّا عند تشغيلها. في مثالنا الذي أوردناه في الشكل ٢-١، الكمبيوتر المادي في المستوى الخامس مثالٌ على ذلك. بطبيعة الحال، كل أنواع عتاد الكمبيوتر عبارة عن أدوات حوسبية مادية.

لكن بعض الأدوات الحوسبية «مجردة» بالكامل. وهي لا تعالج بنيات رمزية فحسب، بل هي في ذاتها بنيات رمزية وخالية في جوهرها من أي خصائص مادية (وإن كان

بالإمكان رؤيتها عبر وسيط مادي مثل العلامات على ورقة أو على شاشة الكمبيوتر). ومن ثَم لا تنطبق عليها القوانين الفيزيائية والكيميائية. فهي لا تشغل حيزًا فيزيائيًّا ولا تستهلك وقتًا فيزيائيًّا. وهي «لا تعمل ولا تدور» في الزمكان الفيزيائي؛ بل توجد بالأحرى في الإطار الخاص بها في الزمكان. لا توجد أمثلة على الأدوات المجردة في الشكل ٢-١. وسأورد بعض الأمثلة في القسم التالي، وسأناقش بعضها في الفصول التالية. لكن إذا كنتم تتذكرون ما قلته عن الإجراءات التي يمكن أن أحدِّدها بصفتي مستخدمًا لأداة «النص» أو «البريد الإلكتروني» لاستخدام هذه الأدوات؛ فهذه الإجراءات أمثلة على الأدوات المجردة.

الفئة الثالثة من الأدوات الحوسبية هي أكثر ما يضفي صفةَ «الغرابة» على علم الكمبيوتر. فهي مجردة ومادية. بعبارة أدق، هي في حد ذاتها بنياتٌ رمزية، وبهذا المعنى تصبح مجردة، لكن تشغيلها يسبِّب تغييرات في العالم المادي: نقل إشارات عبْر مسارات الاتصال، وإشعاع موجات كهرومغناطيسية في الفضاء، ووقوع تغيُّر في الحالات الفيزيائية للأجهزة وهكذا؛ وعلاوة على ذلك، تعتمد أفعالها على كيانٍ مادي كامن لتنفيذ تلك الأفعال. وبسبب هذه الطبيعة، أسميتُ هذه الفئة «الأدوات الحَدِّية» (وتعني كلمة حدِّية هنا حالةً من الغموض أو الوقوف على الحدود بين شيئين). إن برامج الكمبيوتر أو البرمجيات لهي فئة كبيرة من الأدوات الحوسبية الحَدِّية؛ على سبيل المثال، البرامج «النص*» و«البريد الإلكتروني*» و«البحث في الويب*»، وكذلك «نظام التشغيل*» في الشكل ٢-١.

سنقابل نوعًا آخر مهمًّا من الأدوات الحَدِّية لاحقًا. أما في الوقت الحالي، فما يجعل علم الكمبيوتر مميزًا وغريبًا ليس وجود الأدوات الحدِّية فحسب، بل أيضًا إن ما نطلق عليه «الكمبيوتر» هو عبارة عن جهاز يجمع بين الأدوات المادية والمجردة والحدِّية.

على مدارِ ما يقرُب من ستة عقود تطوَّر خلالها علم الكمبيوتر وأصبح مجالًا علميًّا قائمًا بذاته، ظهر العديد من الفئات الفرعية المميزة التي نشأت من هذه الفئات الثلاث للأدوات الحوسبية. يوضح الشكل ٢-١ أربعة أمثلة على ذلك، وهي أداة المستخدِم، والبنية الأساسية، والبرمجيات، والكمبيوتر المادي. بالطبع ترتكز الحوسبة على بعض الفئات الفرعية أكثرَ من غيرها لأنها «أشمل» من غيرها من حيث النطاق والاستخدام. إضافة إلى ذلك، تشكِّل الفئات الأساسية والفئات الفرعية تسلسلًا هرميًّا تركيبيًّا خاصًّا بها.

فيما يلي قائمة ببعض هذه الفئات الأساسية والفرعية المعترَف بها حاليًّا في علم الكمبيوتر. يوضح اصطلاح الترقيم هنا العلاقةَ الهرمية بينها. ونظرًا لأنه قد لا يعرف القارئ العديد من هذه العناصر، فسأشرح أبرزَها في سياق هذا الكتاب.

[١] الأدوات المجرَّدة

[١-١] الخوارزميات
[١-٢] الأوتوماتا المجردة

[١-٢-١] آلات تورنج
[١-٢-٢] الأجهزة التتابعية

[١-٣] اللغات التعريفية
[١-٤] المنهجيات
[١-٥] اللغات

[١-٥-١] لغات البرمجة
[١-٥-٢] لغات وصف العتاد
[١-٥-٣] لغات البرمجة الدقيقة

[٢] الأدوات الحدِّية

[٢-١] أدوات وواجهات المستخدم
[٢-٢] معماريات الكمبيوتر

[٢-٢-١] معماريات المعالج الأحادي
[٢-٢-٢] معماريات المعالج المتعدد
[٢-٢-٣] معماريات الكمبيوتر الموزَّعة

[٢-٣] البرمجيات (البرامج)

[٢-٣-١] أسلوب فون نيومان
[٢-٣-٢] الأسلوب الوظيفي

[٣] الأدوات المادية

[٣-١] أجهزة الكمبيوتر المادية/العتاد
[٣-٢] الدوائر المنطقية
[٣-٣] شبكات الاتصالات

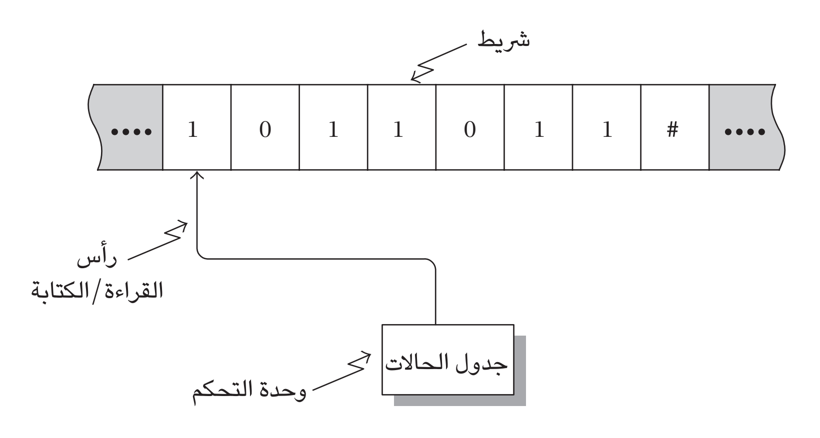
«الآلة الجامعة العظيمة»

توجد أداة حوسبية يجب أن تُميَّز عن غيرها. إنها آلة تورنج، وهي جهاز مجرد يُنسب اسمه إلى صانعه وهو عالِم المنطق والرياضيات ونظريات الكمبيوتر آلان تورنج. اسمحوا لي أن أصفَ هذه الآلة، أولًا ثم أشرح لماذا تستحق هذا الاهتمام الخاص.

تتكوَّن آلة تورنج من شريط غير محدود من حيث الطول ومقسم إلى مربعات. يمكن أن يحمل كل مربع رمزًا واحدًا فقط. وفي أي لحظة من الزمن، يستقر رأس القراءة/الكتابة على مربع واحد من الشريط، والذي يصبح هو المربع «الحالي». الرمز في المربع الحالي (بما في ذلك رمز «الفراغ» أو «المسافة») هو «الرمز الحالي». ويمكن أن تكون الآلة في حالة واحدة ضمن عدد محدود من الحالات. حالة الآلة في أي وقت معيَّن هي «الحالة الحالية». وبناءً على الرمز الحالي والحالة الحالية، يمكن لرأس القراءة/الكتابة أن تكتب رمزًا على المربع الحالي (المخرَج) (بحيث يحل محلَّ الرمز الحالي)، أو تحرِّك أحد المربعات جهة اليمين أو اليسار، أو تحدِث تغييرًا في الحالة فتسمَّى «الحالة التالية». وتتكرَّر دورة التشغيل مع الحالة التالية مثل الحالة الحالية، ويحمل المربع الحالي الجديد رمزًا حاليًّا جديدًا. ويحدِّد «جدول الحالات» العلاقةَ بين الحالات الحالية (المحتملة)، ورموز المدخلات الحالية (المحتملة)، ورموز المخرجات (المحتملة)، وحركات رأس القراءة/الكتابة، والحالات التالية (المحتملة). وسلوك الآلة يحكمه جدول الحالات والآلية غير المرئية التي تؤثِّر في عمليات القراءة والكتابة، وتحرِّك رأس القراءة/الكتابة، وتُحدِث تغييرات في الحالة.

يصوِّر الشكل ٢-٢ آلةَ تورنج بسيطة للغاية تقرأ سلسلة مدخلات مكوَّنة من العددين 0 و1، مكتوبة على شريط، وتستبدل بها أخرى مكوَّنة كلها من العدد 0 إلا في حالة فحص السلسلة بكاملها، فتكتب العدد 1 إذا كان عدد أعداد 1 في السلسلة فرديًّا وتكتب العدد 0 إذا كان غير ذلك. بعد ذلك «تتوقَّف» الآلة. ويشير وجود رمز خاص على الشريط — ولنقل إنه الرمز # مثلًا — إلى نهاية سلسلة المدخلات. يمكن أن يُطلق على هذه الآلة اسم «مكتشف التكافؤ»: فهي تستبدل بسلسلة المدخلات بكاملها أصفارًا وتستبدل بالرمز # العدد 0 أو 1 بناءً على ما إذا كان تكافؤ (عدد أعداد 1 في) سلسلة المدخلات فرديًّا أم زوجيًّا.

تحتاج هذه الآلة ثلاث حالات، أولها، الحالة So وتدُل على اكتشاف عدد فردي من العدد 1 في سلسلة المدخلات في أي وقت من تشغيل الآلة. وثانيها الحالة Se وتمثل اكتشاف عدد زوجي من العدد 1 في أي وقت من تشغيل الآلة. وثالثها الحالة H وهي حالة



شكل ٢-٢: البنية العامة لآلة تورنج.

التوقف؛ أي إنها تؤدي إلى توقُّف الآلة. وعندما يبدأ عمل الآلة، تشير رأس القراءة/الكتابة إلى المربَّع الذي يحمل الرقم الأول في سلسلة المدخلات.

السلوك المحتمَل من آلة تورنج يحدِّده جدول الحالات (انظر الجدول ٢-١).

جدول ٢-١: جدول الحالات

الحالة الحالية	رمز المدخلات	الحالة التالية	رمز المخرجات	حركة رأس القراءة/الكتابة
Se	0	Se	0	اليمين
Se	1	So	0	اليمين
Se	#	H	0	–
So	0	So	0	اليمين
So	1	Se	0	اليمين
So	#	H	1	–

يحدِّد كل صف في هذا الجدول عمليةً مميزة من جانب الآلة، ولا بد من تفسيره، كلٌّ على حدة. على سبيل المثال، يقول الصف الأول ما يلي: إذا كانت الحالة الحالية هي Se ورمز الإدخال الحالي هو 0، فإن الحالة التالية (أيضًا) ستكون Se ويُكتب رمز الإدخال

0 على الشريط وتتحرَّك رأس القراءة/الكتابة موضعًا واحدًا جهة اليمين. ويخبرنا الصف الأخير أنه إذا كانت الحالة الحالية هي So ورمز الإدخال هو #، فستحل القيمة 1 محلَّ رمز # وتصبح الحالة التالية هي حالة التوقف H. ولا يكون ثمَّة حركة أخرى لرأس القراءة/الكتابة.

لنفترض أن سلسلة المدخلات كما هي موضحة في الشكل ٢-٢ وأنه تم ضبط الآلة على الحالة Se. يمكن للقارئ التثبُّت بسهولة من أن تسلسل الحالات ومحتويات الشريط في الدورات المتتالية من تشغيل الآلة سيكونان بالشكل التالي. يشار إلى موضع رأس القراءة/الكتابة في كل دورة بعلامة النجمة على يمين رمز الإدخال «الحالي»:

$$Se\colon 1^*011011\# \longrightarrow\ So\colon 00^*11011\# \longrightarrow So\colon 001^*1011\#$$

$$\longrightarrow\ Se\colon 0001^*011\# \longrightarrow So\colon 00000^*11\#$$

$$\longrightarrow\ So\colon 000001^*1\# \longrightarrow Se\colon 0000001^*\#$$

$$\longrightarrow\ So\colon 0000000\#^* \longrightarrow H\colon 00000001^*$$

إذن، ستكون ثمة آلة تورنج مستقلة (ببساطة أطلق تورنج ذاته عليها اسم «آلة حوسبة») لكل مهمة مستقلة من مهام معالجة الرموز. ستحدد كل آلة تورنج (ذات غرض خاص) أبجدية الرموز التي ستتعرف عليها الآلة، ومجموعة الحالات المحتملة، والمربع الأولي الذي ستُوضع عنده رأس القراءة/الكتابة، وجدول الحالات، والحالة الحالية الأولية. وفي نهاية تشغيل الآلة (حينما تصل إلى حالة «التوقُّف» إن وُجدت)، تعطي المخرجات المكتوبة على الشريط نتيجة مهمة معالجة الرموز.

وبذلك يمكن مثلًا بناء آلة تورنج لجمع العددين n وm، والممثلة بالرمز n متبوعًا بمجموعة من أعداد الواحد ويتبعهما مسافة، ثم يتبع ذلك الرمز m متبوعًا بمجموعة من أعداد الواحد، بحيث يُترك ناتج جمع $n + m$ (في صورة السلسلة $n + m$ ثم مجموعة أعداد الواحد) على الشريط. آلة تورنج أخرى لها سلسلة واحدة مكوَّنة من الرموز a وb وc في صورة مدخلات ستستبدل بسلسلة المدخلات هذه «صورة منعكسة» (والتي تسمَّى «صورة متناظرة») منها. على سبيل المثال، إذا كانت سلسلة المدخلات هي $aaabbbccc$، فإن المخرجات ستكون $cccbbbaaa$. وبذلك تصبح آلة تورنج آلةً لمعالجة الرموز. وهي

بالطبع أداة مجردة على نحوٍ تام، حيث إن الآلة نفسها عبارة عن بنية رمزية. ولم يكن أحد يحلُم بصنع نسخة مادية من آلة تورنج في صورة أداة «عملية».

لكن تورنج زاد على ذلك. فقد أوضح أن بإمكان المرء بناء جهاز حوسبة — ولنقل الجهاز U — يمكن أن يحاكي كل آلات تورنج الأخرى. إذا توافر للجهاز U شريط يحتوي على وصف جدول الحالات لآلة تورنج بعينها، فسيترجم الجهاز U ذلك الوصف وينفِّذ المهمة ذاتها التي ستنفِّذها تلك الآلة بعينها. وبذلك يُطلق على الجهاز U «آلة تورنج العامة».

تكمُن أهمية اختراع تورنج في زعمه بأن «أي إجراء نراه على أنه إجراء حوسبي بطريقة «بديهية» أو «طبيعية» هو إجراء يمكن إنجازه بآلة تورنج». ويترتب على ذلك أن آلة تورنج العامة يمكنها تنفيذُ أي شيء نعتبر أنه حوسبة. يطلق على ذلك الزعم «أطروحة تورنج» (أو يطلق عليه في بعض الحالات أطروحة تشرتش-تورنج؛ حيث إن عالِم منطق آخر وهو ألونزو تشرتش توصَّل إلى الاستنتاج ذاته باستخدام خط تفكير مختلف تمامًا).

يمكننا أن نعتبر آلة تورنج بمثابة «الآلة الجامعة العظيمة». إنها الجامعة لكل الأدوات الحوسبية؛ بمعنى أنه يمكن «اختزال» كل الأدوات الحوسبية وسلوكها فيما تنجزه آلة تورنج.

في ظلِّ ما سبق، وبعد الاعتراف بوجود فرع كامل من علم الكمبيوتر يسمَّى «نظرية الأوتوماتا»، يدرُس بنيةَ آلة تورنج وسلوكها، وقدراتها وحدودها بجميع المظاهر التي يمكن تصوُّرها (مثل حصر الشريط بطول محدَّد أو إدخال عدة أشرطة لها عدة رءوس للقراءة/الكتابة)، لا بد من الاعتراف أيضًا بالموقف المناقض المتمثل في أن آلة تورنج لم يكن لها تقريبًا تأثيرٌ في اختراع أي أداة حوسبية عملية (أو قابلة للاستخدام) ولا في تصميمها ولا تنفيذها ولا سلوكها بأي شكل من الأشكال، وكذلك لم يكن لها تأثير في تفكير علماء الكمبيوتر الذين يتعاملون مع هذه الأدوات ولا في ممارساتهم!

الحوسبة التفاعلية

إضافة إلى ذلك، ظهر منذ زمن تورنج أدوات حوسبية تعمل بعضها مع بعض أو مع أنظمة أخرى طبيعية أو اصطناعية بطريقة تفاعلية. يشير «التفاعل» هنا إلى التأثير المتبادل أو التبادلي فيما بين «الكيانات» الاصطناعية (بما في ذلك الاجتماعية) و/أو الطبيعية بحيث تشكِّل بعضها مع بعض نظامًا من نوعٍ ما.

لنضرب المَثل بدفعي فاتورةَ أحد المرافق؛ ينطوي هذا على تفاعلي وجهاز الكمبيوتر المحمول الخاص بي مع نظام كمبيوتر البنك الخاص بي ونظام كمبيوتر شركة المرافق. في هذا الموقف، تؤثِّر أربعة كيانات (ثلاث أدوات حوسبية وأنا) في عمليات نقل المعلومات والعمليات الحوسبية بطريقة تفاعلية عن طريق تبادل الرسائل والأوامر والبيانات.

أو لنضرب المَثل بالأداة الحوسبية المجرَّدة «النص» في الشكل ٢-١. تشكِّل هذه الأداة واجهةً بين الإنسان والكمبيوتر حيث يتفاعل المستخدِم البشري لأداة «النص» والنظام البرمجي «النص*» بعضهما مع بعض. والأوامر التي توفِّرها أداة «النص» ويصدرها المستخدم تجعل أداة «النص*» تستجيب (مثل بدء سطر جديد في النص، وإنشاء مسافة بين الكلمات، وإضافة حروف لتكوين كلمات في النص، وإنشاء مسافة بدء فقرة جديدة، وكتابة كلمة بخط مائل، وغير ذلك)، واستجابة أداة «النص*» تحفِّز — بدورها — الاستجابةَ من جانب المستخدم البشري.

ولا تمتثل هذه الأنظمة التفاعلية للفكرة «القياسية» لآلة تورنج؛ حيث إن آلة تورنج عبارة عن أداة مستقلة لها مدخلات منقوشة بالفعل على الشريط قبل تشغيل الآلة ولا تُرى مخرجاتها إلا عندما تتوقَّف. أما الأدوات الحوسبية التفاعلية (مثل نظام البنك الخاص بي أو نظام شركة المرافق) فقد لا تتوقَّف عن العمل مطلقًا.

ونظرًا إلى هذه الاعتبارات، يشدِّد بعض علماء الكمبيوتر على أن دراسة آلات تورنج — نظرية الأوتوماتا — حريٌّ بها أن تنتمي إلى علم الرياضيات والمنطق الرياضي أكثرَ من علم الكمبيوتر، وعلى الجانب الآخر يشكِّك آخرون في صحة أن أطروحة تورنج تشمل الحوسبة بكاملها.

علم الكمبيوتر باعتباره علمًا اصطناعيًّا

تلخيصًا لما تناولناه حتى الآن، الأدوات الحوسبية عبارة عن أشياء «مصنوعة»؛ فهي تعالج البنيات الرمزية التي تدُل على المعلومات أو البيانات أو المعرفة (بناءً على وجهة نظر المرء والسياق). وعلم الكمبيوتر هو علم الأدوات الحوسبية.

غنيٌّ عن البيان أن الأدوات الحوسبية ليست جزءًا من العالم الطبيعي بالمعنى الذي تدُل عليه الصخور والمعادن والحفريات، والنباتات والحيوانات، والنجوم والمجرَّات والثقوب السوداء والجسيمات الأولية والذرات والجزيئات. بل الإنسان هو مَن يوجِد هذه الأدوات. وعليه، فعلم الكمبيوتر ليس علمًا «طبيعيًّا». إذن، أيُّ نوع من العلوم هو؟

يقول أحد الآراء، ما دامت الأدوات الحوسبية نفعيةً ومن ثَم تكنولوجية، فإن علم الكمبيوتر ليس علمًا «بحق» على الإطلاق. بل هو فرع من علم الهندسة. ومع ذلك، فإن العلوم الهندسية التقليدية مثل ميكانيكا المواد ونظرية الإنشاءات والديناميكا الحرارية والمعادن الفيزيائية ونظرية الدوائر الكهربية، وكذلك العلوم الهندسية الحديثة مثل الهندسة الحيوية والهندسة الوراثية هي كلها مقيدة مباشرةً بقوانين الطبيعة. وتبدو الأدوات الحوسبية الحدِّية والمجردة بعيدةً كلَّ البعد عن الأدوات المادية الصرفة — كالإنشاءات وأدوات الماكينات والمحركات والدوائر المتكاملة والمعادن والسبائك والمواد المركَّبة وغيرها — التي درسها علماء الهندسة. هذا من أسباب انتماء الأدوات الحوسبية المادية (عتاد الكمبيوتر) في الغالب إلى نطاق مدارس الهندسة، في حين أن الأدوات الحوسبية الحدية والمجردة تنتمي إلى مدارس العلم.

ومع ذلك، هناك شيء مشترك بين جميع الأدوات، سواء الهندسية أو الحوسبية: وهو أنها كلها نتاج لأفكار الإنسان وأهدافه واحتياجاته ورغباته. «تكون الأدوات ذات غرض محدَّد؛ فهي تبرز أهداف صانعيها».

يطلِق هيربرت سايمون على كل العلوم المعنية بالأدوات (المجردة أو الحدية أو المادية) اسم «العلوم الاصطناعية». وهي تتميَّز عن العلوم الطبيعية لأنها يجب ألا تغفل جانب الأهداف والأغراض. فالشيء الطبيعي ليس له غرض: بمعنى أن الصخور والمعادن، والنجوم والمجرَّات، والذرات والجزيئات، والنباتات والكائنات الحية لم تأتِ إلى الكون لغرضٍ بعينه. إنها «موجودة» وحسب. فعالِم الفلك لا يسأل: «ما الهدف من المجرة؟» ولا يسأل عالِم الجيولوجيا: «ما الغرض من الصخور النارية؟» إن مهمة عالِم الطبيعة أن يكتشف القوانين التي تحكم بِنى الظواهر الطبيعية وسلوكها، ويتساءل كيف أتت إلى الوجود، ولكن ليست مهمته أن يسأل لماذا — لأي غرض — أتت إلى الوجود.

على الجانب الآخر، أُوجِدت الأدوات في العالم مجسِّدة لاحتياجات الإنسان وأهدافه. ولا يكفي أن نسأل ما القوانين والمبادئ التي تحكم هيكلَ الأداة الحوسبية وسلوكها (أو — في هذا الصدد — الأهرامات والجسور المعلَّقة ومسرِّعات الجسيمات وسكاكين المطبخ) إذا تجاهلنا عندئذٍ سببَ وجودها.

تنطوي العلوم الاصطناعية على دراسة «العلاقة بين الوسائل والغايات»: بمعنى الأهداف أو الاحتياجات المرادة من الأداة، والأداة التي صُنعت لتلبية الاحتياجات. إذن، تشير كلمة «علم» في علم الكمبيوتر إلى علم الوسائل والغايات. وبناءً على أحد احتياجات

الإنسان أو أهدافه أو أغراضه، فإنها تطرح السؤال: كيف تتمكَّن الأداة الحوسبية من تحقيق ذلك الهدف بشكل قابل للإثبات؟ بمعنًى آخر، كيف يمكن أن يبرهن المرء — من خلال التفكير المنطقي أو الملاحظة أو التجربة — أن الأداة الحوسبية تلبي ذلك الغرض؟

الفصل الثالث

التفكير الخوارزمي

كمثل الشخصية في مسرحية موليير التي لم تكن تعرف أنها تتحدَّث الشِّعر طوال حياتها، قد لا يدرك الكثيرون أنهم كانوا ينفِّذون «خوارزمية» وهم أطفال حينما كانوا يحلون مسألةَ ضرْب أعداد متعددة الأرقام أو يحلون مسألة قسمة مطوَّلة. في الحقيقة، ربما كان الوضع قبل ستينيات القرن العشرين أن القليل ممن هم من خارج أوساط الحوسبة والرياضيات كانوا يعرفون كلمة «خوارزمية». لكن منذ ذلك الحين، وكما هو الحال مع كلمة «النموذج الفكري» (وهو مصطلح ظهر في المجالات النخبوية لفلسفة العلم)، وجدت كلمة «خوارزمية» طريقها إلى اللغة الدارجة كي تعني الصيغ أو القواعد أو الوصفات أو الإجراءات المنهجية لحل المسائل. هذا يرجع إلى حد كبير إلى الارتباط الوثيق — في العقود الخمسة الماضية أو نحو ذلك — بين الحوسبة والخوارزميات.

لكن «مفهوم» الخوارزمية (إن لم يكن الكلمة) يعود إلى العصور القديمة. أورد الكتاب الرائع «الأصول» (الذي ظهر حوالي ٣٠٠ قبل الميلاد)، والذي أصَّل فيه إقليدس لمبادئ الهندسة المستوية، خوارزمية لإيجاد العامل المشترك الأكبر لعددين صحيحين موجبين. نشأت كلمة «خوارزمية» نفسها في القرن التاسع نسبةً إلى عالِم الرياضيات والفلك العربي محمد بن موسى الخوارزمي، الذي عاش وعمِل في أحد المراكز العلمية الرائدة في عصره، وهو بيت الحكمة في بغداد. وفي واحدة من أطروحات الخوارزمي العديدة في الرياضيات وعلم الفلك، كتب عن «فن الحساب الهندي». القرَّاء اللاحقون للترجمات اللاتينية الذين نسبوا هذا العمل خطأً إلى الخوارزمي أطلقوا على عمله كلمة algorismi، وقد أصبحت في النهاية algorism وتعني الإجراء التدرُّجي. ثم تحوَّلت الكلمة

إلى algorithm وتعني الخوارزمية. وأول إشارة وجدها قاموس أكسفورد للغة الإنجليزية لهذه الكلمة كانت في مقالٍ نُشر في دورية علمية إنجليزية عام ١٦٩٥.

دونالد كنوث (الذي لربما كان باعه أطولَ من غيره في جعل الخوارزميات جزءًا من وعي عالِم الكمبيوتر) وصف في إحدى المرات علم الكمبيوتر بأنه دراسة الخوارزميات. لن يتفق كل علماء الكمبيوتر على هذا الرأي، ولكن لا يتصور أحدٌ علم الكمبيوتر من دون أن تكون الخوارزميات في صلب موضوعاته. إنها تشبه نظرية التطور التي وضعها داروين في علم الأحياء؛ إذ يبدو أن كل الطرق في الحوسبة تؤدي إلى الخوارزميات. وإذا كان التفكير من وجهة نظر أحيائية يعني التفكير من وجهة نظر تطوريَّة، فالتفكير من وجهة نظر حوسبيَّة يعني اكتسابَ عادة التفكير الخوارزمي.

اختبار ورقة دَوَّار الشمس

كمدخل لهذا المجال، لنتأمل «اختبار ورقة دَوَّار الشمس» الذي هو من أولى التجارب التي ينفِّذها الطالب في مادة الكيمياء بالمرحلة الثانوية.

توجد مادة سائلة غير معروفة في أنبوب اختبار أو دورق. يغمس منفِّذ التجربة قطاعًا من ورقة دَوَّار شمس زرقاء في هذه المادة السائلة. إذا تحوَّلت إلى اللون الأحمر، دلَّت على أن المادة السائلة حمضية؛ وإذا بقيت باللون الأزرق، دلَّت على أن المادة السائلة ليست حمضية. في الحالة الثانية، يغمس منفِّذ التجربة قطاعًا من ورقة دَوَّار شمس حمراء في المادة السائلة. وإذا تحوَّلت إلى اللون الأزرق، دلَّت على أن المادة السائلة قلوية، وإذا لم تتحوَّل كانت المادة السائلة محايدة.

هذا «إجراء تقريري» يتعلَّمه الطلاب في المراحل المبكِّرة جدًّا من دراستهم لمادة الكيمياء، ويمكن وصفه بالطريقة التالية:

```
if a blue litmus strip turns red when dipped into a liquid
      then conclude the liquid is acidic
      else
        if a red litmus strip turns blue when dipped in the liquid
          then conclude the liquid is alkaline
          else conclude the liquid is neutral
```

سيرِد الترميز المستخدم هنا كثيرًا في هذا الفصل وبعض الفصول التالية، وحريٌّ بنا توضيحه. بوجه عام، يُستخدم الترميز If C then $S1$ else $S2$ في التفكير الخوارزمي لسرد خطوات اتخاذِ قرارٍ ما. إذا تحقَّق الشرط C، فإن «تدفُّق التحكم في الخوارزمية» ينتقل إلى المقطع $S1$، وحينها سينفَّذ المقطع $S1$. لكن إذا لم يتحقَّق الشرط C، فسينتقل التحكم إلى المقطع $S2$، وحينها سينفَّذ المقطع $S2$. وفي كلتا الحالتين، بعد تنفيذ العبارة if then else، ينتقل التحكم إلى العبارة التالية في الخوارزمية.

لاحظ أن منفِّذ التجربة ليس بحاجة إلى معرفة أي شيء عن سبب عمل اختبار ورقة دَوَّار الشمس بالطريقة يعمل بها. هو ليس بحاجة إلى معرفة ماهية «ورقة دَوَّار الشمس» — أي تركيبتها الكيميائية — ولا العملية الكيميائية التي تتسبَّب في تغيير لون الورقة. لتنفيذ التجربة، يكفي تمامًا أن يتعرَّف منفِّذ التجربة على ورقة دَوَّار الشمس عند رؤيتها، وأن يمكنه ربط تغيُّر لون الورقة بالأحماض أو القلويات.

أصبح مصطلح «اختبار ورقة دَوَّار الشمس» استعارةً تدُل على حالة محددة أو اختبار حاسم. ولأسباب وجيهة، فإن نجاح هذه الطريقة مضمون. «ستكون» ثمة نتيجة لا لبس فيها، ولا سبيل إلى وجود الشك. إضافة إلى ذلك، لا يستغرق اختبار ورقة دَوَّار الشمس أجلًا غير مسمًّى؛ فمنفِّذ التجربة على يقين بأن الاختبار سيعطي نتيجةً بعد «مدة محددة من الزمن».

هذه الخصائص المجمَّعة لاختبار ورقة دَوَّار الشمس — الذي هو إجراء آلي مضمون أنه سيعطي نتيجة صحيحة في مدة زمنية محددة — هي العناصر الأساسية التي تميِّز الخوارزميات.

متى يكون الإجراء خوارزمية؟

في نظر علماء الكمبيوتر، الخوارزمية ليست مجرد وصفة أو إجراء آلي. وحتى يتأهل الإجراء إلى مرتبة الخوارزمية حسب فهْم علماء الكمبيوتر لهذا المفهوم، فلا بد أن تتوافر به السِّمات التالية (التي كان دونالد كنوث أولَ مَن حدَّدها):

المحدودية. لا بد للخوارزمية من نهاية (بمعنى أنها تتوقف) بعد عدد محدد من الخطوات.

الوضوح الشديد. لا بد من تحديد كل خطوة في الخوارزمية تحديدًا دقيقًا لا لبس فيه.

الفاعلية. كل عملية تنفَّذ كجزء من الخوارزمية يجب أن تكون بسيطة لدرجة تمكِّن الإنسان من تنفيذها بالضبط (باستخدام ورقة وقلم على سبيل المثال).

توافر المُدخلات والمُخرجات. يجب أن تتضمن الخوارزمية مُدخلًا أو أكثر وكذلك مُخرجًا أو أكثر.

لنضرب المَثل بخوارزمية إقليدس الجليلة المذكورة سابقًا لإيجاد العامل المشترك الأكبر للعددين الصحيحين الموجبين m وn (أي أكبر عدد صحيح موجب يقبل القسمة الصحيحة على العددين m وn). ترِد الخوارزمية هنا بلغةٍ تجمع بين اللغة العادية والرموز الرياضية الأولية وبعض الرموز المستخدَمة للدلالة على النتائج (كما هو الحال في مثال اختبار ورقة دَوَّار الشمس). في الخوارزمية، m وn هما «متغيرا مدخلات» وn أيضًا هي «متغير مخرجات». إضافة إلى ذلك، يلزم وجود «متغير مؤقت» ثالث يُرمز إليه بالرمز r. وكذلك يوضع «التعليق»، الذي ليس جزءًا من الخوارزمية نفسها، بين القوسين {}. وللرمز ⟵ أهمية خاصة في الخوارزمية: فهو يدل على «عملية التعيين»؛ بمعنى أن $a \leftarrow b$ يُقصد به نسخ أو تعيين قيمة المتغير a إلى المتغير b.

Euclid's GCD algorithm. Given two positive integers m and n find their GCD.

Input m, n $\{m, n \geq 1\}$;

Temp var r;

***Step 1*:** divide m by n; $r \leftarrow$ remainder; $\{0 \leq r \leq\}$;

Step 2*: *if $r = 0$ **then Output** n;

Halt

else

***Step 3*:** $m \leftarrow n$;

$n \leftarrow r$

goto step 1.

لنفترض مبدئيًّا أن m يساوي 16 وn يساوي 12. إذا «نفَّذ» شخص هذه الخوارزمية باستخدام القلم والورقة، فإن قيم المتغيرات الثلاثة m وn وr بعد تنفيذ كل خطوة ستكون بالشكل التالي:

	m	n	r
Step 1:	16	12	4;
Step 3:	12	4	4;
Step 1:	12	4	0;
Step 2:	Output $n = 4$.		

مثال آخر، لنفترض مبدئيًّا أن m يساوي 17 وn يساوي 14. ستكون قيم المتغيرات الثلاثة بعد تنفيذ كل خطوة كما يلي:

	m	n	r
Step 1:	17	14	3;
Step 3:	14	3	3;
Step 1:	14	3	2;
Step 3:	3	2	2;
Step 1:	3	2	1;
Step 3:	2	1	1;
Step 1:	2	1	0;
Step 2:	Output $n = 1$.		

في المثال الأول، العامل المشترك الأكبر للعددين (16، 12) هو 4، وهو مُخرج الخوارزمية حينما تنتهي؛ وفي المثال الثاني، العامل المشترك الأكبر للعددين (17، 14) هو 1، وهو مُخرج الخوارزمية بعدما تنتهي.

من الواضح أن الخوارزمية لها مُدخلات. الأمر الأقل وضوحًا هو ما إذا كانت الخوارزمية ستفي بمعيار المحدودية أم لا. فالخوارزمية تتضمن تكرارًا أو إعادة يمليها

الأمر goto ما يجعل التحكم يرجع إلى الخطوة الأولى. وكما هو واضح من المثالين، تتكرَّر الخوارزمية بين الخطوتين الأولى والثالثة حتى يتحقق الشرط $r = 0$، وعليه تصبح قيمة n المخرج وحينها تنتهي الخوارزمية. يبيِّن المثالان بوضوح أنه بالنسبة إلى هذه الأزواج المعينة من قيم المدخلات m وn، فدائمًا ما تفي الخوارزمية ب «معيار الإنهاء» $(r = 0)$ وستتوقف حينها. لكن كيف نعرف ما إذا كانت أزواج القيم الأخرى لن تظل تتكرر إلى ما لانهاية وتتبدل بين الخطوتين الأولى والثالثة وتؤدي إلى عدم إنتاج مخرجات البتة؟ (في هذا الموقف، ستخرق الخوارزمية كلًّا من معيار المحدودية ومعيار توافر المخرجات.) وكيف نعرف أن الخوارزمية ستفرغ من العمل دومًا فيما يتعلق بكل القيم الموجبة الممكنة للمتغيرين m وn؟

الإجابة هي أنه يجب التأكد من أن الخوارزمية لها نهاية بوجه عام. يكمُن هذا في أنه بعد كل اختبار للشرط $r = 0$ في الخطوة الثانية، تكون قيمة r أقل من العدد الصحيح الموجب n، وقيمتا n وr تقلان مع كل تنفيذ للخطوة الأولى. لا بد أن يصل التسلسل المتناقص للأعداد الصحيحة الموجبة في النهاية إلى القيمة 0، وفي النهاية يتحقق الشرط $r = 0$، ومن ثَم ينتهي الإجراء في نهاية المطاف بمقتضى الخطوة الثانية.

ماذا عن معيار الوضوح الشديد؟ ينص هذا المعيار على ضرورة تحديد كل خطوة في الخوارزمية تحديدًا دقيقًا. ويجب تحديد الإجراءات المراد تنفيذها بما لا يدع مجالًا للغموض. وهنا تظهر «اللغة» في الصورة. يستخدم وصف خوارزمية إقليدس مزيجًا من اللغة والرموز الرياضية التي يشوبها الغموض. والشخص الذي ينفِّذ تلك الخوارزمية عقليًّا (باستخدام الورقة والقلم) من المفترض أن يفهم معنى القسمة والباقي والأعداد الصحيحة فهمًا دقيقًا. لا بد أن يفهم معنى الترميز الصوري أكثر، مثل الرموز if…then…else وgoto.

أما بشأن الفاعلية، فيجب أن تكون كل العمليات التي سيجري تنفيذها بسيطةً لدرجة أنه يمكن تنفيذها في مدة زمنية محدودة. في هذه الحالة الخاصة، العمليات مبسطة لدرجة أنه يمكن تنفيذها على ورقة مثلما جرى.

«المضي قُدُمًا وأداء عملية الضرب»

يرتبط مفهوم «التجريد» بمواصفات الخوارزميات. بعبارة أخرى، يمكن حل مسائل معينة باستخدام خوارزميات محددة على مستويين مختلفين أو أكثر من مستويات التجريد.

قبل اختراع حاسبات الجيب، كان الأطفال يتعلمون مسائل الضرب باستخدام الورقة والقلم. وما يلي هو ما تعلمته في صغري. ولأغراض التبسيط، لنفترض أن عددًا مكوَّنًا من ثلاثة أرقام (المضروب) يُضرب في عدد مكوَّن من رقمين (المضروب فيه).

الخطوة ١: نكتب العددين بحيث يكون المضروب في الصف العلوي والمضروب فيه في الصف السفلي، ونحاذي رقم آحاد المضروب فيه بحيث يكون تحت رقم آحاد المضروب بالضبط.

الخطوة ٢: نرسم خطًّا أفقيًّا تحت المضروب فيه.

الخطوة ٣: نضرب المضروب في رقم آحاد المضروب فيه ونكتب الناتج (حاصل الضرب الجزئي) تحت الخط الأفقي، ونكتب الناتج بحيث نحاذي كل أرقام الآحاد.

الخطوة ٤: نضع 0 تحت رقم آحاد حاصل الضرب الجزئي الذي حصلنا عليه من الخطوة الثالثة.

الخطوة ٥: نضرب المضروب في رقم عشرات المضروب فيه ونكتب الناتج (حاصل الضرب الجزئي) في الصف الثاني تحت الخط الأفقي على يسار الرقم 0.

الخطوة ٦: نرسم خطًّا أفقيًّا آخر تحت حاصل الضرب الجزئي الثاني.

الخطوة ٧: نجمع حاصلَي الضرب الجزئيَّين، ونكتب ناتج الجمع تحت الخط الأفقي الثاني.

الخطوة ٨: تنتهي المسألة. فالعدد تحت الخط الأفقي الثاني هو حاصل الضرب المطلوب.

لاحظ أن تنفيذ هذا الإجراء يتطلب أن يكون لدى الطفل بعض المعرفة المسبقة: أولًا، يجب أن يعرف طريقة ضرْب عدد مكوَّن من عدة أرقام في عدد مكوَّن من رقم واحد. ينطوي هذا على ضرورة حفظ جدول الضرب لعددين يتكوَّن كلٌّ منهما من رقم واحد، أو إمكانية الوصول إلى جدول الضرب. ثانيًا، لا بد أن يعرف طريقة جمع الأعداد المكوَّنة من رقمين أو أكثر، ويجب أن يعرف طريقة التعامل مع مسائل الجمع بالحمل. ثالثًا، يجب أن يعرف طريقة جمع عددين مكوَّنين من عدة أرقام.

مع ذلك، ليس الطفل بحاجةٍ إلى معرفةِ أو فهْمِ «السبب» في محاذاة العددين طبقًا للخطوة الأولى؛ ولا «سبب» نقلِ حاصل الضرب الجزئي الثاني موضعًا واحدًا جهةَ اليسار طبقًا للخطوة الخامسة، ولا «سبب» إدخال 0 في الخطوة الرابعة، ولا «سبب» الحصول على حاصل الضرب الصحيح بعد جمع حاصلَي الضرب الجزئيين في الخطوة السابعة.

لكن لاحظ دقة الخطوات. فما دام يتَّبع الشخص هذه الخطوات بالضبط كما هو مذكور، فسيكفل ذلك نجاحَ الإجراء بشرط توافر الشرطين الأول والثاني المذكورين فيما سبق في الشخص الذي ينفِّذ هذا الإجراء. من المؤَّكد أن ينتج عن هذه الخطوات النتيجة المطلوبة في مدة زمنية محدودة: وهذا من السمات الأساسية في الخوارزمية.

لنتفكر كيف ينفِّذ معظمنا عمليةَ الضرب في هذه الأيام. سنُخرج حاسبة الجيب (أو الهاتف الذكي) من جيوبنا، وسنبدأ في استخدام الحاسبة كما يلي:

الخطوة ١: نُدخل المضروب.

الخطوة ٢: نضغط على علامة ×.

الخطوة ٣: نُدخل المضروب فيه.

الخطوة ٤: نضغط على علامة =.

الخطوة ٥: نتوقَّف. يظهر حاصل الضرب على الشاشة.

هذه أيضًا خوارزمية ضرب. تعطي الخوارزميتان حاصل الضرب ذاته ولكنهما على مستويين مختلفين من التجريد. ولا يعرف المستخدِم ما الذي يحدث تحديدًا عند تنفيذ الخطوات من الأولى إلى الرابعة في الخوارزمية الثانية. ربما الحاسبة تنفِّذ الخوارزمية ذاتها التي تنفَّذ بالورقة والقلم. وربما أيضًا تستخدم طريقةَ تنفيذ مختلفة. فالمستخدم لا يرى هذه المعلومات.

مستويات التجريد في هذا المثال تنطوي أيضًا على مستويات من «الجهل». فالطفل الذي يستخدم خوارزمية الورقة والقلم يعرف معلومات عن عملية الضرب أكثرَ من الشخص الذي يستخدم حاسبة الجيب.

ثبات الخوارزميات

تمتاز الخوارزميات بخاصية مريحة وهي أن أداءها لا يعتمد على منفِّذها، ما دام توافر في المنفِّذ شروط المعرفة الثلاثة التي سبق ذكْرها. فما دام لم تتغير المدخلات إلى الخوارزمية، فلن تتغير المخرجات بغض النظر عمن (أو ما) ينفِّذها. إضافة إلى ذلك، ستعطي الخوارزمية الناتج ذاته بغض النظر عن وقت تنفيذها. وبوجه عام، تشير هاتان السمتان إلى أن الخوارزميات تتسم بخاصية الثبات.

هذا يفسِّر عدم اعتبار الوصفات المذكورة في كتب الطهي ضمن الخوارزميات؛ فكثيرًا ما تتضمَّن هذه الوصفات خطوات غامضة، ومن ثَم تقوِّض معيار الوضوح الشديد. على سبيل المثال، قد تتضمَّن هذه الوصفات تعليمات لإضافة مكونات «مهروسة قليلًا» أو «مبشورة بشرًا ناعمًا» أو نصيحة «للطهي على نار هادئة». هذه التعليمات بالغة الغموض ولا تفي بالشروط التي ينبغي توافرها في الخوارزمية. بل إنها تُترك لحَدْس الطاهي وخبرته وحكمه كي يفسِّرها. وهذا يفسِّر سببَ اختلاف مذاق الطبق ذاته الذي أعدَّه طاهيان مختلفان بالوصفة ذاتها، أو لماذا يختلف مذاق الوصفة ذاتها التي أعدَّها شخص واحد في مناسبتين مختلفتين. فهذه الوصفات تخلُّ بمبدأ الحسم.

الخوارزميات أدوات مجردة

لا شك أن الخوارزمية أداة؛ فقد صمَّمها الإنسان أو ابتكرها كي يحقق أهدافه أو احتياجاته. وبقدرِ ما أنها تعالج البنيات الرمزية (كما في حالة خوارزميتي الضرب والعامل المشترك الأكبر)، فإنها أدوات حوسبية. (ليست جميع الخوارزميات تعالج البنيات الرمزية؛ فاختبار ورقة دَوَّار الشمس يتطلَّب كيانات مادية — أنبوب سائل الاختبار، وشريحة ورقة دَوَّار الشمس — باعتبارهما المدخلات، وينتج حالة فيزيائية — لون شريحة ورقة دَوَّار الشمس — باعتبارها المخرجات. إن اختبار ورقة دَوَّار الشمس عبارة عن خوارزمية يدوية تعمل بناءً على كيانات فيزيائية وكيميائية وليس بنيات رمزية؛ ومن ثَم لا يجدُر بنا اعتبارها أداة حوسبية.)

لكن الخوارزميات في حد ذاتها ليس لها وجود مادي، سواء أكانت حوسبية أم غير ذلك. فالإنسان لا يستطيع لمسها أو إمساكها أو تحسُّسها أو تذوُّقها أو سماعها. وهي لا تخضع لقوانين علوم الفيزياء ولا الكيمياء ولا حتى قوانين الهندسة. بل هي «أدوات مجردة». إنها تتألَّف من البنيات الرمزية التي — شأنها شأن كل البنيات الرمزية — تمثِّل أشياء في العالم، وقد تكون هذه الأشياء مادية (كورقة دَوَّار الشمس، والمواد الكيميائية، والأزرار في حاسبة الجيب، وغيرها) أو مجردة (كالأعداد الصحيحة، والعمليات المنفذة على الأعداد الصحيحة، والعلاقات مثل التساوي، وغير ذلك).

الخوارزمية أداة مساعدة. وكما هو الحال مع معظم الأدوات المساعدة، كلما قل احتياج المستخدِم إلى معرفة المفاهيم النظرية في الخوارزمية، زادت فاعليتها بالنسبة إليه.

هذا يثير النقطة التالية: يمكن اعتبار الخوارزمية أداة لها وجهان مثل يانوس. (يانوس هو إله البوابات الروماني الذي كان ينظر في اتجاهين معاكسين في آن واحد.) يتطلب «تصميم» الخوارزمية أو «ابتكارها» إبداعًا في العموم، ولكن «استخدامها» عبارة عن عمل ميكانيكي صِرف ويتطلب القليل من التفكير الإبداعي. وإن جاز التعبير، فالخوارزمية شكلٌ من أشكال التفكير غير الواعي.

الخوارزميات معرفة «إجرائية»

الخوارزميات أدواتٌ ينشئها المستخدِمون لحل المسائل. وبمجرد صياغتها وإتاحتها للعامة، تصبح ملكًا للعالم. هذا ما يجعل الخوارزميات أدواتٍ موضوعية (وكذلك ثابتة). لكن الخوارزميات تُعَد تجسيدًا للمعرفة أيضًا. وحيث إنها أدوات موضوعية، فهي تجسيدٌ لما يطلق عليه فيلسوف العلم كارل بوبر «المعرفة الموضوعية». (قد يبدو متناقضًا إذا قلنا إن مستخدم الخوارزمية «مفكِّر غير واعٍ» وفي نفس الوقت «كائن عالم»، ولكن حتى التفكير غير الواعي يظل تفكيرًا، والتفكير يستلزم البناء على المعرفة بأحد أشكالها.) لكن ما نوع المعرفة التي تمثِّلها الخوارزمية؟

في العلوم الطبيعية، نتعلم التعريفات والحقائق والنظريات والقوانين وغير ذلك. وفيما يلي بعض الأمثلة من مبادئ الفيزياء والكيمياء.

(١) سرعة الضوء المتَّجهة في فراغ تساوي 186 ألف ميل في الثانية.
(٢) التسارع هو معدَّل تغيُّر السرعة المتجهة.
(٣) الوزن الذري للهيدروجين يساوي 1.
(٤) للمادة أربع حالات وهي الصلبة والسائلة والغازية والبلازما.
(٥) عند تنظيم العناصر الكيميائية حسب ترتيب أعداد الذرات، فإنه يوجد نمط دوري (متكرر) لخصائص العناصر.
(٦) الاحتراق يتطلب وجود الأكسجين.

كل جملة «تُقرر» شيئًا ما: أن الاحتراق يتطلب وجود الأكسجين؛ وأن الوزن الذري للهيدروجين يساوي 1؛ وأن التسارع هو معدَّل تغيُّر السرعة المتجهة؛ وهكذا. وصحة هذه الجمل (أو التقدير التقريبي لذلك) يكون إما عن طريق التعريف (كما في المثال الثاني)

أو الحساب (كما في المثال الأول) أو التجربة والملاحظة (كما في المثالين الرابع والسادس) أو التفكير المنطقي (كما في المثال الخامس). في الواقع، عند النظر إلى تلك الجمل على نحوٍ منفصل، فإنها تُعَد عناصر المعلومات التي تصبح جزءًا من معرفة الشخص عند استيعابها (ارجع إلى الفصل الأول). هذا النوع من المعرفة يسمَّى «المعرفة التقريرية» أو باللغة الدارجة المعرفة «النظرية».

تحتوي الرياضيات أيضًا على معرفة تقريرية، في صورة تعريفات أو مسلَّمات أو مبرهنات. على سبيل المثال، يُعَد مبدأ الاستقراء الرياضي مسلَّمةً أساسية في الحساب وضعها عالِم الرياضيات الإيطالي جوزيبي بيانو:

> أي خاصية تنطبق على الصفر وكذلك على العدد التالي مباشرةً لعددٍ تتوافر به هذه الخاصية، تنطبق على كل الأعداد.

وعلى النقيض من ذلك، تندرج مبرهنة فيثاغورس ضمن المعرفة التقريرية في الهندسة المستوية عن طريق التفكير المنطقي (البرهان):

> في المثلث القائم الزاوية، مربع الوتر يساوي مجموع مربعَي طولَي الضلعين الآخرين.

> وفيما يلي مثال على المعرفة التقريرية في الرياضيات عن طريق التعريف:
> مضروب العدد الصحيح غير السالب n هو:

$$\text{factorial } (n) = 1 \text{ for } n = 0 \text{ or } n = 1$$

$$= n(n-1)(n-2)\ldots 3.2.1 \text{ for } n > 1$$

على الجانب الآخر، لا تُعدُّ الخوارزمية تقريرية؛ بل هي إجراء يوضح كيفية إنجاز شيءٍ ما. إنها تصف عملًا من نوعٍ ما. وعليه، فإن الخوارزمية مثال على «المعرفة الإجرائية»، أو بالمعنى الدارج المعرفة «العملية».

وبالنسبة إلى عالِم الكمبيوتر، لا يكفيه أن يعرف أن مضروب العدد يُعرَّف بكذا وكذا. فهو يريد أن يعرف طريقةَ حساب مضروب عددٍ ما. بعبارة أخرى، هو بحاجة إلى خوارزمية. على سبيل المثال:

FACTORIAL

Input: $n \geq 0$;

Temp variable: *fact;*

Step 1: $fact \leftarrow 1$;

Step 2: if $n \neq 0$ **and** $n \neq 1$ **then**

repeat

Step 3: $fact \leftarrow fact * n$;

Step 4: $n = n - 1$;

Step 5: until $n = 1$;

Step 6: output *fact;*

Step 7: halt

الترميز Repeat *S* until *C* يحدِّد «تكرارًا» أو «حلقة تكرارية». وهذا يعني أنه سيتكرَّر تنفيذ العبارة (العبارات) *S* إلى أن يتحقق الشرط *C*. وعندما يحدث هذا، تنتهي الحلقة التكرارية ويتدفَّق التحكم إلى العبارة التي تلي التكرار.

في الخطوة الأولى من هذا المثال، عُينت القيمة 1 إلى *fact* (المضروب). وإذا كانت قيمة n تساوي 0 أو 1، فلن يتحقق الشرط في الخطوة الثانية، وفي هذه الحالة سينتقل التحكم إلى الخطوة السادسة مباشرةً وتصبح قيمة المضروب 1 وتنتهي الخوارزمية عند الخطوة السابعة. على الجانب الآخر، إذا كانت قيمة n ليست 0 أو 1، يتكرر تنفيذ «الحلقة التكرارية» المشار إليها بالمقطع Repeat...until وفي كل مرة تتناقص قيمة n بمقدار 1 حتى يتحقَّق «شرط إنهاء» الحلقة التكرارية $n = 1$. عندئذٍ ينتقل التحكم إلى الخطوة السادسة التي عند تنفيذها تعطي قيمةَ المضروب بالشكل التالي: $n(n-1)(n-2)\ldots 3.2.1$.

لاحظ أنه يمكن تقديم المفهوم ذاته — المضروب — بطريقة تقريرية (كما يفضِّل علماء الرياضيات) وبطريقة إجرائية (كما يحب علماء الكمبيوتر). في الحقيقة، يطرح الشكل التقريري «النظرية» الأساسية («ما المضروب؟») للشكل الإجرائي أو الخوارزمية («كيف نحسب المضروب؟»).

باختصار، تعبِّر الخوارزميات عن شكلٍ من أشكال المعرفة الإجرائية التي تتسم بالموضوعية أيضًا.

تصميم الخوارزميات

عندما نفكِّر في «تصميم» الخوارزميات، فإن تجريدها له تبِعات مثيرة للفضول. يرجع السبب في ذلك بوجه عام إلى أن التصميم عملٌ هادف (ذو غرض محدَّد) يبدأ بمجموعة من المتطلبات R المراد تحقيقها باستخدام أداة A لم تُنشأ بعد، وينتهي ببنية رمزية تمثِّل الأداة المطلوبة. في الحالات العادية، تكون البنية الرمزية هذه هي التصميم $D(A)$ الخاص بالأداة A. ويكون هدف المصمِّم هو إنشاء هذا التصميم بحيث إذا نُفِّذت تلك الأداة طبقًا له فإنها ستفي بالمتطلبات R.

لا ينطوي هذا السيناريو على مشاكل حينما تكون الأداة A مادية؛ فتصميم جسر على سبيل المثال سيكون تمثيلًا لهيكل الجسر في صورة رسومات هندسية ومجموعة من العمليات الحسابية والمخطَّطات التي توضِّح القوى العاملة على الهيكل. لكن في حالة اعتبار الخوارزميات كأدوات، فإن الأداة ذاتها بنيةٌ رمزية. ومن ثَم يصبح الحديث عن تصميم خوارزمية حديثًا عن بنية رمزية (التصميم) التي تمثِّل بنيةً رمزية أخرى (الخوارزمية). وهذا ينطوي على قدرٍ من الحيرة.

وهكذا في حالة الخوارزميات يكون من المنطقي والمعقول أكثر اعتبار التصميم والأداة شيئًا واحدًا. فمهمة تصميم خوارزمية هي مهمة إنشاء بنية رمزية تكون هي الخوارزمية A بحيث إن A تحقِّق المتطلبات R.

الكلمة المهمة في هذا المقام هي «إنشاء». إن عملية التصميم عملٌ إبداعي، وكما أوضح الباحثون في مجال الإبداع، فالعمل الإبداعي مزيجٌ معقَّد من الإدراك والمنطق والحَدْس والمعرفة وحسن التقدير والدهاء والاكتشافات التي تتم مصادفة. ومع ذلك يتحدَّث واضعو نظريات التصميم عن «علم التصميم» أو «منطق التصميم».

فهل يوجد إذن جانبٌ علمي في تصميم الخوارزميات؟ الإجابة هي: «إلى حدٍّ ما». توجد ثلاثة أوجه أساسية يدخل بها «المنهج العلمي» في تصميم الخوارزميات.

بادئ ذي بدء، لا تأتي مسألة التصميم من فراغ. فهي تؤطَّر في سياق كمٍّ من المعرفة (سنطلق عليه «فضاء المعرفة») ذي صلة بالمسألة ويحوزها المصمم. ولدى تصميم خوارزمية جديدة، يصبح فضاء المعرفة هذا ذا صلة. على سبيل المثال، قد يُكتشف وجه تشابه بين المسألة المطروحة والمسألة التي حُلت بخوارزمية موجودة حاليًّا (والتي هي جزء من فضاء المعرفة)، ومن ثَم يمكن نقل الأسلوب المطبَّق في المسألة الثانية إلى المسألة الراهنة. وهذه إحدى حالات «التفكير القياسي». أو ربما تبدو استراتيجية تصميم معروفة

مناسبة للمسألة القائمة على وجه الخصوص، ومن ثَم يمكن تجربة هذه الاستراتيجية وإن كان بغير ضمان على نجاحها. وهذه إحدى حالات «التفكير الاستدلالي». أو ربما توجد نظرية صورية ذات صلة بالمجال الذي تنتمي إليه المسألة؛ ومن ثَم يمكن أخذُ النظرية في الاعتبار. وهذه إحدى حالات «التفكير النظري».

بعبارة أخرى، قد يكون لأشكال التفكير المختلفة تأثيرٌ في تصميم الخوارزمية بناءً على كمٍّ من المعرفة المقرَّرة أو المجرَّبة أو المثبَتة (والتي تكون تقريرية وإجرائية على حد سواء). سنطلق على ذلك اسم «عامل المعرفة» في تصميم الخوارزمية.

لكن مجرد التوصُّل إلى الخوارزمية ليس كافيًا. فهناك أيضًا ضرورة إقناع النفس والآخرين بأن الخوارزمية «صالحة». وتنطوي هذه الضرورة على اتباع التفكير المنهجي في توضيح أن الخوارزمية تستوفي المتطلبات الأصلية. سأطلق على ذلك اسم «عامل الصلاحية» في تصميم الخوارزميات.

وأخيرًا، وحتى لو ثبَت أن الخوارزمية صالحة، فربما لا يكون هذا كافيًا. فثمَّة مسألة أداء الخوارزمية: ما مدى كفاءة الخوارزمية؟ وسنطلق على ذلك اسم «عامل الأداء» في تصميم الخوارزمية.

تنطوي هذه «العوامل» الثلاثة على أنواع التفكير والمنطق وقواعد البرهان التي نربطها عادةً بالعلم. فلنتناول بشيء من الأمثلة كيفيةَ إسهامها في علم تصميم الخوارزميات.

مشكلة تحويل التعبيرات الحسابية

توجد فئةٌ من برامج الكمبيوتر تسمَّى «البرامج المترجِمة» ووظيفتها تحويل البرنامج المكتوب بلغة برمجة عالية المستوى (وهي لغة تكون مستقلة عن سمات أجهزة الكمبيوتر المادية الفعلية، مثل لغة فورتران أو سي++) إلى سلسلة تعليمات يمكن لأجهزة كمبيوتر مادية خاصة تنفيذها (تفسيرها) مباشرةً. ويطلق على سلسلة التعليمات الخاصة بالآلة اسم «لغة الآلة». (سنتناول لغات البرمجة في الفصل الرابع.)

واجه كتَّاب البرامج المترجِمة الأوائل (في أواخر خمسينيات القرن العشرين وستينياته) مشكلةً تقليدية وهي صياغة خوارزميات تحوِّل «التعبيرات الحسابية» التي تظهر في البرنامج إلى لغة آلة. ومن الأمثلة على ذلك التعبير:

$$(a + b) * (c - 1/d)$$

في هذا المثال، العلامات + و− و∗ و/ هي العوامل الحسابية الأربعة؛ ويطلق على المتغيرات a وb وc وd والرقم الثابت 1 اسم «المعاملات». كذلك يطلق على التعبير بهذه الصيغة والذي تظهر فيه العوامل الحسابية بين معاملَيْه اسم «التعبير المرتب وسطيًّا».

وفضاء المعرفة الذي يحيط بتلك المسألة (ويحوزه مصمم الخوارزمية) يتضمَّن «قواعد الأسبقية» التالية الخاصة بالعوامل الحسابية:

(أ) في حالة غياب الأقواس، تكون الأسبقية لعاملَي الضرب ∗ والقسمة / على عاملَي الجمع + والطرح −.

(ب) عاملا الضرب ∗ والقسمة / يستويان في مستوى الأسبقية، وعاملا الجمع + والطرح − يستويان في مستوى الأسبقية.

(ج) إذا ظهر في التعبير عوامل ذات مستوى أسبقية واحد، يصبح ترتيب الأسبقية من اليسار إلى اليمين. بمعنًى آخر، تطبَّق العوامل على المعاملات بالترتيب الموجودة به من اليسار إلى اليمين.

(د) التعبيرات الموجودة داخل الأقواس لها أعلى درجات الأسبقية.

من ثَم، في التعبير الوارد فيما سبق على سبيل المثال، سيكون ترتيب العوامل بالشكل التالي:

(أ) نفِّذ $a + b$. وأطلق على النتيجة $t1$.
(ب) نفِّذ $1/d$. وأطلق على النتيجة $t2$.
(ج) نفِّذ $c - t2$. وأطلق على النتيجة $t3$.
(د) نفِّذ $t1 * t3$.

على الجانب الآخر، إذا كان التعبير خاليًا من الأقواس كما يلي:

$$a + b * c - 1/d$$

فإن ترتيب العوامل سيكون بالشكل التالي:

(١) نفِّذ $b * c$. وأطلق على النتيجة $\mathrm{t1}'$.
(٢) نفِّذ $1/d$. وأطلق على النتيجة $\mathrm{t2}'$.
(٣) نفِّذ $a + t1'$. وأطلق على النتيجة $\mathrm{t3}'$.
(٤) نفِّذ $\mathrm{t3}' - \mathrm{t2}'$.

يمكن تصميم خوارزمية لإنشاء لغة آلة بحيث تقيِّم عند تنفيذها التعبيرات الحسابية المرتبة وسطيًّا تقييمًا صحيحًا طبقًا لقواعد الأسبقية. (ستعتمد الطبيعة المحددة للخوارزمية على طبيعة التعليمات المعتمدة على الآلة، والتي تختلف حسب جهاز الكمبيوتر المادي المحدد.) ومن ثَم وبناءً على قواعد الأسبقية، تستعين الخوارزمية بقواعدَ دقيقة هي جزء من فضاء المعرفة ذات الصلة بالمسألة. كذلك ولأن الخوارزمية تعتمد على قواعد الأسبقية اعتمادًا مباشرًا، فإن إثبات صلاحية الخوارزمية سيصبح يسيرًا إلى حد بعيد. لكن وكما توضِّح الأمثلة السابقة، تزيد الأقواس إلى حدٍّ ما من تعقيد عملية تحويل التعبير المرتَّب وسطيًّا.

وثمَّة ترميزٌ لتحديد التعبيرات الحسابية من دون الحاجة إلى الأقواس من ابتكار عالِم المنطق البولندي يان ووكاسيفيتش (١٨٧٨–١٩٥٦)، ومن ثَم أصبح معروفًا باسم «الترميز البولندي». يطلق على إحدى صيغ هذا الترميز «الصيغة البولندية العكسية»، وفيها يأتي العامل الحسابي بعد المعاملَين مباشرةً في صورة تعبير بولندي عكسي. توضح الأمثلة التالية الصيغةَ البولندية العكسية لبعض التعبيرات المرتَّبة وسطيًّا.

(أ) بالنسبة إلى التعبير $a + b$، الصيغة البولندية العكسية هي $ab+$.
(ب) بالنسبة إلى التعبير $a + b - c$، الصيغة البولندية العكسية هي $ab + c-$.
(ج) بالنسبة إلى التعبير $a + b * c$، الصيغة البولندية العكسية هي $a\ b\ c*+$.
(د) بالنسبة إلى التعبير $(a + b) * c$، الصيغة البولندية العكسية هي $a\ b + c*$.

يتقدَّم تقييم التعبير البولندي العكسي من اليسار إلى اليمين بطريقة مباشرة، ومن ثَم يزيد من سهولة مسألة الترجمة. تنص القاعدة على أن العوامل الحسابية الواردة في التعبير تنطبق على معاملاتها السابقة عليها حسب ترتيب ظهور العوامل، من اليسار إلى اليمين. على سبيل المثال، في حالة التعبير المرتَّب وسطيًّا:

$$(a + b) * (c - 1/d)$$

الصيغة البولندية العكسية هي:

$$ab + c1d/ - *$$

وترتيب التقييم هو:

(١) نفِّذ $ab+$ وأطلق على النتيجة $t1$. ومن ثَم يكون التعبير الناتج هو $t1\ c\ 1d/-*$.
(٢) نفِّذ $1\ d/$ وأطلق على النتيجة $t2$. ومن ثَم يكون التعبير الناتج هو $t1\ c\ t2-*$.
(٣) نفِّذ $c\ t2-$ وأطلق على النتيجة $t3$. ومن ثَم يكون التعبير الناتج هو $t1\ t3*$.
(٤) نفِّذ $t1\ t3*$.

بالطبع سيكتب المبرمجون التعبيرات الحسابية بالصيغة المرتبة وسطيًّا المعتادة. وسينفذ البرنامج المترجِم خوارزمية ستحوِّل تلك الصيغ إلى صيغة بولندية عكسية أولًا، ثم يولِّد لغة آلة من التعبيرات البولندية العكسية.

مسألة تحويل التعبيرات من الصيغة المرتبة وسطيًّا إلى الصيغة البولندية العكسية توضِّح مدى إمكانية اجتماع أساس نظري سليم واستراتيجية تصميم مجربة أثناء تصميم خوارزمية يمكن إثبات صحتها.

يطلق على استراتيجية التصميم «الاستدعاء الذاتي»، وهي حالة خاصة من استراتيجية أشمل لحل المسائل تُعرف باسم «فرِّق تسُد». في الاستراتيجية الثانية، في المسألة P، إذا كان يمكن تقسيم المسألة إلى مسائل فرعية أصغر $p1$، و$p2$، و... pn، فجد حل كل مسألة فرعية على حدة، ثم اجمع حلول المسائل الفرعية للحصول على حل المسألة الأساسية P.

في الاستدعاء الذاتي، تنقسم المسألة P إلى عدد من المسائل الفرعية من النوع ذاته مثل المسألة P ولكنها أصغر. ثم تنقسم المسألة الفرعية إلى مسائلَ فرعية أصغر من النوع ذاته، وهكذا إلى أن تصبح المسائل الفرعية صغيرةً وبسيطة لدرجة أنه يمكن حلها مباشرةً. عندئذٍ تُجمع حلول المسائل الفرعية لإيجاد حلول المسائل الفرعية «الأكبر»، ثم تُجمع حلول المسائل الفرعية الأكبر لإيجاد حلول المسائل الأكبر حتى نصل إلى حل المسألة الأصلية P.

انظر الآن في مسألة تحويل التعبيرات من الصيغة المرتبة وسطيًّا إلى الصيغة البولندية العكسية في الخوارزميات. وهذه المسألة قائمة على مجموعة من القواعد الصورية:

لنفترض أن $B = \{+, -, *, /\}$ هي مجموعة العوامل الحسابية الثنائية (بمعنى أن كل عامل b في B له معاملان بالضبط). ولنفترض أن a يرمز إلى أحد المعاملات. ولنفترض

أيضًا أنه إذا كان تعبير ذو صيغة مرتبة وسطيًّا هو I فستكون صيغته البولندية العكسية هي I'. إذن:

(أ) إذا كان I معاملًا فرديًّا هو a، فستكون الصيغة البولندية العكسية هي a.
(ب) إذا كان $I1\ b\ I2$ تعبيرًا مرتبًا وسطيًّا وكان b أحد عناصر B، فإن التعبير البولندي العكسي المقابل له هو $I1'I2'b$.
(ج) إذا كان (I) تعبيرًا مرتبًا وسطيًّا، فستكون صيغته البولندية العكسية هي I'.

تظهر الخوارزمية الذاتية الاستدعاء المكوَّنة مباشرةً من هذه القواعد بعد ذلك في صورة «دالة» — بالمعنى الرياضي للكلمة. في الرياضيات، الدالة F التي تطبق على قيمة «المُدخل» x ويُرمز إليها بالدالة Fx أو $F(x)$ تُنتج قيمة الدالة للمُدخل x. على سبيل المثال، الدالة المثلثية الخاصة بجيب الزاوية التي تطبق على المُدخل 90 (درجة)، ويرمز إليها كما يلي: SIN90، تنتج القيمة 1. كذلك دالة الجذر التربيعي التي يُرمز إليها بالرمز $\sqrt{\ }$ وتطبق على مُدخل، ولنقل 4 (ومن ثَم تصبح $\sqrt{4}$)، تنتج القيمة 2.

وبناءً على ذلك، فإن الخوارزمية المسماة هنا RP التي مُدخلها التعبير المرتب وسطيًّا I تصبح بالشكل التالي:

RP (I)

Step 1: **if** $I = a$ **then** return a
else
Step 2: **if** $I = I1\ b\ I2$
then return RP $(I1)$ RP $(I2)$ b
else
Step 3: **if** $I = (I1)$ **then** return RP $(I1)$
Step 4: halt

في الخطوة الثالثة، الصيغة العامة if C then S حالة خاصة من صيغة القرار If then else: بمعنى ألا يتدفق التحكم إلى S إلا إذا تحقَّق الشرط C، وإلا فسيتدفق التحكم إلى العبارة التي تلي if then.

من ثَم يمكن للدالة RP أن تنشِّط نفسها على نحوٍ ذاتي الاستدعاء باستخدام قيم مُدخلات «أصغر». ولا يخفى أن دالة RP هي تنفيذ مباشر لقواعد التحويل، ومن ثَم فهي صحيحة من حيث التركيب. (بالطبع ليست كل الخوارزميات صحيحة بهذا الوضوح؛ قد يكون أساسها النظري أكثر تعقيدًا بكثير، ويجب حينها إثبات صحتِها من خلال حجة دقيقة أو حتى شكل من أشكال البراهين الرياضية؛ أو قد يكون أساسها النظري ضعيفًا أو لا وجود له حتى.)

لتوضيح كيف تعمل الخوارزمية مع المدخلات الفعلية، انظر الأمثلة التالية:

(a) Suppose $I = a + b$. Then:

$RP\ (a + b) = RP\ (a)\ RP\ (b) +$	(by step 2)
$= ab +$	(by step 1 twice)

(b) Suppose $I = (a + b) * c$. Then:

$RP\ ((a + b) * c) = RP\ (a + b)\ RP\ (c)*$	(by step 2)
$= RP\ (a)\ RP\ (b) + RP\ (c)*$	(by step 2)
$= ab + c*$	(by step 1 thrice)

(C) Suppose $I = (a * b) + (c - 1/d)$. Then:

$RP\ ((a * b) + (c - 1/d))$

$= RP\ (a * b)\ \ RP\ (c - 1/d) +$	(by step 2)
$= RP\ (a)\ RP\ (b) * RP\ (c)\ RP\ (1/d) - +$	(by step 2 thrice)
$= RP\ (a)\ RP\ (b) * RP\ (c)\ RP\ (1)\ RP\ (d)/ - +$	(by step 2)
$= ab * c1d/- +$	(by step 1 five times)

كفاءة الخوارزميات باعتبارها أدوات نفعية

قلنا سابقًا إن الأمر لا يقتصر على تصميم خوارزمية صحيحة. وكما هو الحال مع مصمم أي أداة نفعية، يجب أن يهتم مصمم الخوارزمية بمدى «كفاءة» هذه الخوارزمية؛ أي، مدى فاعليتها في القيام بمهامها. فهل يمكننا «قياس» كفاءة الخوارزمية بناءً على هذا

المعنى؟ هل يمكننا عملُ مقارنة نوعية بشكلٍ ما بين خوارزميتين متنافستين تؤديان المهمة ذاتها؟

عامل الكفاءة الجليُّ هو مقدار الوقت الذي تستغرقه الخوارزمية في تنفيذ المهمة. لكن الخوارزمية أداة مجرَّدة. فلا يمكننا قياسها بالزمن الفعلي؛ لا يمكننا قياس الزمن على ساعة حقيقية؛ حيث إن الخوارزمية كخوارزمية لا تتضمن أيَّ شيء مادي. فإذا كنت أنا الإنسان أنفِّذ خوارزمية، فيُفترض أن بإمكاني قياسَ مقدار الوقت الذي استغرقته في تنفيذ الخوارزمية ذهنيًّا (ربما باستخدام الورقة والقلم). لكن هذا ليس إلا قياسًا «لأدائي أنا» للخوارزمية بناءً على مجموعة معينة من المعطيات. واهتمامنا ينصبُّ على قياس أداء الخوارزمية عبْر كل مُدخلاتها المحتملة وبغضِّ النظر عمن ينفِّذ الخوارزمية.

لكن في نهاية المطاف، يفترض مصمِّمو الخوارزميات أن كل خطوة أساسية في الخوارزمية تستغرق وحدةَ الزمن ذاتها. اعتبر هذه الوحدة «زمنًا مجردًا». كذلك فهم يتصورون حجم المسألة التي صُمِّمت الخوارزمية من أجلها من حيث عدد عناصر البيانات المعنية بها المسألة. حينها يعتمدون مقياسين لـ «كفاءة» الخوارزميات. المقياس الأول له صلة بأسوأ أداء محتمل للخوارزمية باعتبارها دالة لحجم المسألة n؛ ويتعلَّق المقياس الثاني بمتوسط الأداء في صورة دالة لحجم المسألة n. ويطلق على المقياسين اسم «تعقيد الوقت». (وهناك مقياس بديل وهو تعقيد المساحة: والمقصود به مقدار مساحة الذاكرة (المجردة) اللازمة لتنفيذ الخوارزمية.)

متوسط تعقيد الوقت هو مقياس الكفاءة الأكثر واقعية، ولكنه يتطلب استخدام الاحتمالات، ومن ثَم تزيد صعوبة تحليله. وفي هذا المقام، لن نتناول غير سيناريو أسوأ الحالات.

انظر المسألة التالية. لديَّ قائمة تضم العدد n من العناصر. وكل عنصر يحتوي على اسم طالب وعنوان بريده الإلكتروني. والقائمة مرتبة أبجديًّا حسب الاسم. تدور المسألة حول البحث في القائمة وإيجاد عنوان البريد الإلكتروني لاسم طالب بعينه.

أبسط طريقة هي البدء من عند رأس القائمة ومطابقة كل جزء من الاسم بالعنصر المعطَى من اسم الطالب، والتقدُّم عبْر القائمة اسمًا باسم حتى نجد تطابقًا، ثم نخرج عنوان البريد الإلكتروني المقابل له. (لغرض التبسيط، سنفترض أن اسم الطالب المعطى في مكانٍ ما بالقائمة.) يُطلق على هذه الخوارزمية اسم «خوارزمية البحث الخطي».

LINEAR SEARCH

Input: *student:* an array of n entries, each entry consisting of two 'fields', denoting *name* (a character string) and *email* (a character string) respectively. For the i-th entry in *student,* denote the respective fields by student *[i].name* and *student[i].email.*

Input: *given-name:* the name being 'looked up'.

Temp variable *i:* an integer

Step 1: i ← 1;
Step 2: ***while*** *given-name ≠ student[i].name*
Step 3: **do** i ← I + 1;
Step 4: output *student[i].email*
Step 5: halt

في هذا المثال، الترميز العام While C do S يحدِّد شكلًا جديدًا من التكرار ويعني أنه بما أن الشرط C تحقَّق، كرر تنفيذ العبارة («متن الحلقة التكرارية») S. وعلى النقيض من الترميز Repeat S until C، يُختبر شرط الحلقة التكرارية قبل إدخال متنها في كل تكرار.

في سيناريو أسوأ حالة ممكنة، تظهر الإجابة المطلوبة في الإدخال الأخير (ذات الترتيب n). لذا في سيناريو الحالة الأسوأ، ستتكرر حلقة While التكرارية عدد n من المرات. في هذه المسألة، تكون قيمة n — والتي تمثِّل عدد الطلاب في القائمة — هي العامل الحاسم والمحوري: هذا هو حجم المسألة.

لنفترض أن كل خطوة تستغرق مقدار الوقت ذاته تقريبًا. في أسوأ الحالات، تحتاج خطوات هذه الخوارزمية مقدار الوقت $2n + 3$ = خطوات زمنية حتى تعثر على النتيجة المطابقة. لنفترض أن قيمة n كبيرة للغاية (ولنقل 20 ألفًا). في هذه الحالة، يصبح عامل الجمع 3 جديرًا بالإهمال ويمكن تجاهله. لكن عامل الضرب 2 الذي يضاعف قيمة n عامل ثابت. أما ما يهيمن فهو قيمة n التي هي حجم المسألة؛ هذا هو ما قد تختلف قيمته من قائمة طلاب إلى أخرى. إذن، نحن مهتمون بأن نقول شيئًا عن كفاءة الخوارزمية من حيث مقدار الوقت (المجرد) اللازم لتنفيذ الخوارزمية باعتبارها دالة لقيمة n هذه.

إذا كانت الخوارزمية تعالج مسألةً بالحجم n في المدة الزمنية kn، حيث k ثابت، فإننا نقول إن تعقيد الوقت من الرتبة n، ويُرمز إليه بالرمز $O(n)$. يطلق على هذا

ترميز Big O، والذي قد قدَّمه عالِم الرياضيات الألماني بي باخمان عام ١٨٩٢. ويتيح لنا هذا الترميز طريقةً لتحديد فاعلية (تعقيد) الخوارزمية باعتبارها دالةً لحجم المسألة. وفي حالة خوارزمية البحث الخطي، فإن تعقيدها في أسوأ الحالات يكون $\mathrm{O}(n)$. وإذا كانت الخوارزمية تحل المسألة في أسوأ الحالات في الوقت kn^2، فإن تعقيد الوقت فيها في أسوأ الحالات يساوي $\mathrm{O}(n^2)$. أما إذا كانت الخوارزمية تستغرق الوقت $kn\log n$، فإن تعقيد الوقت فيها يساوي $\mathrm{O}(n\log n)$، وهكذا دواليك.

إذن، يتضح أنه في مسألة واحدة بالحجم n، فإن الخوارزمية ذات التعقيد $\mathrm{O}(\log n)$ تستغرق وقتًا أقل من الخوارزمية ذات التعقيد $\mathrm{O}(n)$، والتي ستستغرق وقتًا أقل من الخوارزمية ذات التعقيد $\mathrm{O}(n\log n)$، وهذه الأخيرة ستستغرق وقتًا أقل من الخوارزمية ذات التعقيد $\mathrm{O}(n^2)$، والتي ستكون أفضل من الخوارزمية ذات التعقيد $\mathrm{O}(n^3)$. أسوأ الخوارزميات هي التي يكون تعقيد الوقت فيها دالةً «أسية» للعدد n، مثل الخوارزمية ذات التعقيد $\mathrm{O}(2^n)$. هذه التباينات في كفاءة الخوارزميات التي تتضمَّن هذه الأنواع من تعقيدات الوقت عرَضها علماء الكمبيوتر ألفريد أهو، وجون هوبكروفت، وجيفري أولمان في كتابهم البارز الصادر عام ١٩٧٤ بعنوان «تصميم الخوارزميات وتحليلها». وعلى افتراض أن خطوات الخوارزمية تستغرق مقدارًا معينًا من الوقت الفعلي، فقد أوضحوا أنه في دقيقة واحدة يمكن للخوارزمية ذات التعقيد $\mathrm{O}(n)$ أن تحل المسائل ذات الحجم $n = 6 * 10^4$ في دقيقة واحدة، والخوارزمية ذات التعقيد $\mathrm{O}(n\log n)$ لنفس المسألة أن تحل المسائل ذات الحجم $n = 4,893$، والخوارزمية ذات التعقيد $\mathrm{O}(n^3)$ أن تحل المسألة ذاتها على أن يكون حجمها $n = 39$، والخوارزمية الأسية ذات التعقيد $\mathrm{O}(2^n)$ أن تحل فقط المسألة ذات الحجم $n = 15$.

وبذلك يمكن وضع الخوارزميات في ترتيب هرمي بناءً على تعقيد وقتها بناءً على Big O، بحيث تأتي الخوارزمية ذات التعقيد $\mathrm{O}(\mathrm{k})$ (حيث يعبِّر k عن ثابت) على رأس التسلسل الهرمي والخوارزميات الأسية ذات التعقيد $\mathrm{O}(\mathrm{k}^n)$ في أدنى التسلسل. وتقل كفاءة الخوارزميات بشكل ملحوظ مع الهبوط في التسلسل الهرمي.

لنفكر في مسألة البحث في قائمة الطلاب، ولكن هذه المرة نراعي حقيقة أن الإدخالات في القائمة مرتبة أبجديًّا حسب اسم الطالب. في هذه المسألة، ربما يفعل الباحث ما يفعله حين يبحث في دليل الهاتف أو حينما يبحث في القاموس. عندما نبحث في دليل الهاتف، فإننا لا نفتح على الصفحة الأولى وننظر في كل اسم على حدة. لكن على افتراض أن الكلمة التي نبحث عن معناها في القاموس تبدأ بحرف K، فإننا نفتح صفحات القاموس حتى

نكاد نصل إلى صفحات الكلمات التي تبدأ بحرف K. وإذا فتحنا القاموس على صفحات الكلمات التي تبدأ بحرف M على سبيل المثال، فإننا نعرف أن علينا أن نقلب الصفحات إلى الخلف؛ وإذا فتحنا على حرف H فسنقلب الصفحات إلى الأمام. إننا نقلِّل مقدارَ البحثِ بفضل ميزة الترتيب الأبجدي.

يمكن اتباع هذا النهج بشكلٍ أكثر دقة باستخدام خوارزمية تسمَّى «البحث الثنائي». لنفترض أن القائمة تحتوي على عدد من الإدخالات قدره $k = 2^n - 1$، ويجري تحديد الإدخال الأوسط في كل خطوة. فإذا تعرَّفت الخوارزمية على اسمٍ يقع «بعد» اسم التلميذ المعطى حسب الترتيب الأبجدي، فستتجاهل الخوارزمية الإدخالات جهةَ اليسار من العنصر الأوسط. وعندئذٍ ستحدد الإدخال الأوسط في النصف الأيمن من القائمة وتعيد المطابقة. وإذا لم تعثر على الاسم في كل مرة، فستعيد تقسيم القائمة إلى نصفين وتستمر على ذلك حتى تعثر على الاسم المطلوب.

لنفترض أن القائمة تضم $k = 15$ (أي، $2^4 -$) من الإدخالات. ولنفترض أن القائمة مرقمة من 1 إلى 15. عندئذٍ يسهل التحقق من أن الحد الأقصى من المسارات التي ستسلكها الخوارزمية سيكون واحدًا مما يلي:

$$8 \longrightarrow 4 \longrightarrow 2 \longrightarrow 1$$

$$8 \longrightarrow 4 \longrightarrow 2 \longrightarrow 3$$

$$8 \longrightarrow 4 \longrightarrow 6 \longrightarrow 5$$

$$8 \longrightarrow 4- \longrightarrow 6 \longrightarrow 7$$

$$8 \longrightarrow 12 \longrightarrow 10 \longrightarrow 9$$

$$8 \longrightarrow 12 \longrightarrow 10 \longrightarrow 11$$

$$8 \longrightarrow 12 \longrightarrow 14 \longrightarrow 13$$

$$8 \longrightarrow 12 \longrightarrow 14 \longrightarrow 15$$

هنا، الإدخال 8 هو الإدخال الأوسط. إذن، لن يتم البحث في أكثر من $\log_2 16 = 4$ من الإدخالات قبل العثور على تطابق. وبالنسبة إلى قائمة بالحجم n، فإن أسوأ أداء لخوارزمية البحث الثنائي هو $O(\log n)$، وهذا يمثِّل تحسُّنًا بالمقارنة مع خوارزمية البحث الخطي.

جماليات الخوارزميات

التجربة الجمالية — البحث عن الجمال — لا توجد في الفن والموسيقى والسينما والأدب فحسب، بل توجد في العلوم والرياضيات وحتى التكنولوجيا. استهل الشاعر جون كيتس الأبيات الأخيرة من قصيدة «قصيدة على جَرة إغريقية» التي كتبها عام ١٨٢٠ بقوله: «الجمال هو الحقيقة، والحقيقة هي الجمال». وقد رفض عالِم الرياضيات الإنجليزي جي إتش هاردي — محاكيًا كيتس — فكرةَ وجود «رياضيات قبيحة» رفضًا تامًّا.

تأمَّل لماذا يسعى علماء الرياضيات إلى إيجاد براهين مختلفة لمبرهنة بعينها. بمجرد أن يتوصَّل عالمٌ إلى برهان على مبرهنة، لماذا يكلِّف عالِم آخر نفسه عناءَ إيجاد برهان آخر مختلف؟ تكمُن الإجابة في أن علماء الرياضيات يبحثون عن براهينَ جديدة على المبرهنات حينما تكون البراهين الحالية غير جذابة من المنظور الجمالي. إنهم يبحثون عن الجمال في الرياضيات.

ينطبق الأمر بالقدْر نفسه على تصميم الخوارزميات. قد يوجد حلٌّ لمسألةٍ ما باستخدام خوارزميةٍ ما ولكنها نوعًا ما قبيحة؛ بمعنى أنها غير متقنة أو تستغرق وقتًا طويلًا. يتجلى هذا القبح في بعض الأحيان في كون الخوارزمية غيرَ فعَّالة. لذا، يبحث علماء الكمبيوتر — لا سيما ذوو الخلفية الرياضية — عن الجمال في الخوارزميات بنفس الشكل الذي يبحث به علماء الرياضيات عن الجمال في البراهين. ربما كان أفصح المتحدثين عن الجمال في الخوارزميات هم علماء الكمبيوتر إدسخر ديكسترا من هولندا وسي إيه آر هور من بريطانيا ودونالد كنوث من الولايات المتحدة. وكما أشار ديكسترا ذات مرة: «الجمال اختصاصنا.»

يمكن إشباع هذه الرغبة في الجمال بالسعي إلى أن تكون الخوارزميات أبسط، أو ذات تركيب أفضل، أو تستخدم مفاهيمَ عميقة.

لنضرب المَثل بخوارزمية المضروب التي سبق ذكرها في هذا الفصل. قامت هذه الخوارزمية التكرارية على تعريف دالة المضروب كما يلي:

$$\text{fact}\ (n) = 1 \text{ for } n = 1 \text{ or } n = 0$$

$$= n\ (n-1)(n-2)\ldots 3.2.1 \text{ for } n > 1$$

لكن يوجد تعريف ذاتي الاستدعاء لدالة المضروب:

$$\text{fact}\ (n) = 1 \text{ for } n = 1 \text{ or } n = 0$$

$$= n * \text{fact}\ (n - 1) \text{ for } n > 1$$

وبذلك تصبح الخوارزمية المقابلة — في صورة دالة — بالشكل التالي:

```
rec-fact (n)
    if n = 0 or n = 1
        then return 1
        else return n * rec-fact (n − 1)
```

سيلمس العديد من علماء الكمبيوتر قدرًا أكبرَ من الجمال في هذه الخوارزمية بفضل صيغتها الواضحة وسهولة فهمها وبساطتها، وكذلك لأنها تستغل التعريف الذاتي الاستدعاء الأدق لدالة المضروب. اعلم أن الخوارزميات ذاتية الاستدعاء وغير ذاتية الاستدعاء (التكرارية) تقف على مستويات مختلفة من التجريد: بمعنى أن الخوارزمية ذاتية الاستدعاء قد ينفِّذها شكل متغير من نظيرتها التي ليست ذاتية الاستدعاء.

المسائل المستعصية الحل («البالغة الصعوبة»)

أنْهِ هذا الفصل بتحويل التركيز من الخوارزميات التي تحل المسائل الحوسبية إلى المسائل الحوسبية ذاتها. في قسمٍ سابق، علِمنا أن أداء الخوارزميات يحدَّد بناءً على تعقيد الوقت (أو المساحة) الخاص بها. على سبيل المثال، في مسألة البحث في قائمة الطلاب، تُبرِز الخوارزميتان (البحث الخطي والبحث الثنائي) تعقيدين مختلفين للوقت في أسوأ الحالات على الرغم من أنهما تحلان «مسألة» واحدة.

لكن لنتأمل المسألة المسماة «مسألة البائع المتجول»: لنفترض أن هناك عددًا من المدن ومسافات الطرق بينها، فهل بإمكان البائع المتجول أن يبدأ بمدينته الأصلية ويزور كل المدن ثم يعود إلى مدينته الأصلية بحيث تكون المسافة المقطوعة أقلَّ من أو تساوي قيمة معينة؟ في الحقيقة، لا توجد خوارزمية معروفة تحل هذه المسألة وتكون ذات تعقيد وقت أقل من التعقيد الأسي $O(k^n)$ حيث يعبِّر k عن ثابت وn عن حجم المسألة (مثل عدد المدن).

يقال إن المسألة الحوسبية «مستعصية الحل» — أي، بالغة الصعوبة — إذا كانت كل الخوارزميات المعروف أنها تحل المسألة تبلغ على الأقل درجةَ تعقيد الوقت الأسي. أما المسائل التي لها خوارزميات ذات «تعقيد وقت متعدد الحدود» (مثل $O(n^k)$ فيقال إنها يمكن حلها؛ بمعنى أنها «ممكنة الحل عمليًّا».

يندرج فرع علم الكمبيوتر الذي يتعامل مع قابلية أو عدم قابلية المسائل الحوسبية للحل ضمن المجالات الرياضية الصورية ويسمَّى «نظرية التعقيد الحوسبي»، والذي تأسس في ستينيات وأوائل سبعينيات القرن العشرين في الغالب على يد العالم الإسرائيلي مايكل رابين والعالم الكندي ستيفن كوك، والعلماء الأمريكيين جوريس هارتمانيس وريتشارد ستيرنس وريتشارد كارب.

يفرِّق منظِّرو التعقيد بين فئتين من المسائل، هما مسائل التعقيد المتعددة الحدود (P) ومسائل التعقيد المتعددة الحدود غير القطعية (NP). التعريفان الصوريان (أي، الرياضيان) لهاتين الفئتين معقَّدان، كما أنهما يتعلقان بنظرية الأوتوماتا، ولا سيما أنواعًا معينة من آلات تورنج (ارجع إلى الفصل الثاني)، ولا حاجة إلى أن يعرقلانا في هذا المقام. ومن المنظور غير الصوري، تتكون الفئة P من كل المسائل التي يمكن حلها في الوقت المتعدد الحدود — ومن ثَم فهذه المسائل يمكن حلها. كذلك من المنظور غير الصوري، تتكون الفئة NP من المسائل التي يمكن التحقق من صحة حلها المقترح في الوقت المتعدد الحدود، وهذا الحل ربما يمكن أو لا يمكن الحصول عليه في الوقت المتعدد الحدود. على سبيل المثال، مسألة البائع المتجول ليس لها حل خوارزمي (معروف) في الوقت المتعدد الحدود، ولكن «بفرض» أن ثمة حلًّا، فيمكن التحقق «بسهولة» مما إذا كان صحيحًا في الوقت المتعدد الحدود أم لا.

لكن مسألة البائع المتجول مستعصية الحل كما ذكرنا. ومن ثَم قد تتضمن الفئة NP مسائلَ يُعتقد أنها مستعصية الحل — على الرغم من أن الفئة NP يندرج ضمنها أيضًا مسائل الفئة P ممكنة الحل.

إن تداعيات هذه الأفكار كبيرة. ويحظى المفهوم «مسائل التعقيد المتعددة الحدود غير القطعية الكاملة» باهتمام خاص. يقال إن المسألة π «مسألة متعددة الحدود غير قطعية كاملة» إذا كانت تقع ضمن الفئة NP وكانت كل المسائل الأخرى في هذه الفئة يمكن «تحويلها» أو «اختزالها» في الوقت المتعدد الحدود إلى π. يعني هذا أنه إذا كانت المسألة π مستعصية الحل، فإن كل المسائل الأخرى في الفئة NP مستعصية الحل. وبالعكس،

إذا كانت المسألة π ممكنة الحل، فإن كل المسائل الأخرى في الفئة NP ممكنة الحل أيضًا. ومن ثَم تكون كل المسائل في الفئة NP «متكافئة» بهذا المعنى.

في عام ١٩٧١، طرح ستيفن كوك مفهومَ المسألة المتعددة الحدود غير القطعية الكاملة، وأثبت أن مسألة بعينها تسمَّى «مسألة قابلية الإرضاء» متعددة الحدود غير قطعية كاملة. (تتضمن مسألة قابلية الإرضاء تعبيرات بولينية (أو منطقية) — مثل التعبير $(a \text{ or } b) \text{ and } c$ — حيث a وb وc عبارة عن متغيرات بولينية (منطقية) لا تحتمل غير القيمتين (الخاصتين بالحقيقة) وهما TRUE وFALSE. تقول المسألة: «هل توجد مجموعة من قيم الحقيقة لحدود تعبير منطقي بحيث تكون قيمة التعبير TRUE؟») أثبت كوك أنه يمكن اختزال أي مسألة ضمن الفئة NP إلى مسألة قابلية الإرضاء التي تقع أيضًا ضمن المسائل الكثيرة الحدود غير القطعية. ومن ثَم إذا كانت مسألة قابلية الإرضاء قابلة للحل/مستعصية الحل، فهكذا ستكون كل المسائل الأخرى في الفئة NP.

ومن هذا برز بعد ذلك السؤال التالي: «هل توجد خوارزميات ذات وقت متعدد الحدود لكل مسائل الفئة NP؟» ذكرنا سابقًا أن هذه الفئة تندرج ضمنها مسائل الفئة P. لكن ما يطرحه هذا السؤال هو: هل مسائل الفئة P «مطابقة» لمسائل الفئة NP؟ هذا هو ما يطلق عليه «مسألة P = NP»، ويمكن القول إنها أشهر مسألة مفتوحة في علم الكمبيوتر النظري. لم يُثبِت أحد أن P = NP، ويعتقد الكثيرون أن هذا ليس الحال؛ أي، إن الكثيرين يعتقدون أن P ≠ NP (لكن لم يتم إثبات ذلك حتى الآن). من شأن هذا أن يعني أنه توجد مسائل في الفئة NP (مثل مسألة البائع المتجوِّل ومسألة قابلية الإرضاء) لا تقع ضمن الفئة P، ومن ثَم فهي مستعصية الحل بطبيعتها، وإذا كانت كاملة ضمن الفئة NP فإن كل المسائل الأخرى القابلة للاختزال إليها تصبح مستعصية الحل أيضًا. هذا يعني أنه لا توجد خوارزميات ممكنة عمليًّا لهذه المسائل.

ما تخبرنا به نظرية مسائل التعقيد الكثيرة الحدود غير القطعية الكاملة هو أن العديد من المسائل التي تبدو متمايزة تكون «مرتبطة بعضها ببعض» بطريقة غريبة. فيمكن تحويل إحداها إلى أخرى؛ فهذه المسائل مكافئة لبعضها. ويتسنى لنا فهمُ أهمية هذه الفكرة بمجرد أن ندرك أن مجموعة كبيرة من المسائل الحوسبية القابلة للتطبيق في مجالات الأعمال والإدارة والصناعة والتكنولوجيا — مشكلات «العالم الواقعي» — هي مسائل كثيرة الحدود غير قطعية كاملة: وإذا ما استطاع أحدهم فقط أن ينشئ خوارزمية فعَّالة (ذات وقت متعدد الحدود) لواحدة من هذه المسائل، فإن بإمكانه إيجاد خوارزمية مجدية لكل المسائل الأخرى.

إذن، «كيف» يتكيف المرء مع هذه المسائل المستعصية على الحل؟ سنتناول أحد النُّهج الشهيرة في هذا الصدد في الفصل السادس.

الفصل الرابع

فن البرمجة وعلمها وهندستها

من باب التذكير، التفكير الخوارزمي أساسي بالنسبة إلى علم الكمبيوتر. إلا أن الخوارزميات أدواتٌ مجردة. بإمكان علماء الكمبيوتر أن يرتضوا العيش (إذا رغبوا في ذلك) في عالَم الخوارزميات العاجي، وألا ينغمسوا مرة أخرى في «العالَم الواقعي»، تمامًا كما قد يفعل علماء الرياضيات «البحتة» إلى حد كبير. لكن إذا أردنا من أجهزة الكمبيوتر المادية الحقيقية أن تنفِّذ المهام الحوسبية نيابةً عنا، أو إذا أردنا من أجهزة الكمبيوتر المادية ألا تكتفيَ بتنفيذ أنواع المهام الحوسبية التي تعيينا كثيرًا (على الرغم من أهميتها) بل تنفذ أيضًا المهام التي تتجاوز قدراتنا المعرفية الطبيعية، فلن يكفي أن نتحلى بالتفكير الخوارزمي وحده. لا بد من «صياغة» هذه الخوارزميات بطريقة تستطيع أجهزة الكمبيوتر المادية أن تفهمها وتفسِّرها وتنفِّذها وفقًا لمعاييرها وليس معايير الإنسان.

من هنا، دخلت البرمجة إلى مسرح الأحداث الحوسبي. برنامج الكمبيوتر عبارة عن توصيف للعملية الحوسبية المرغوبة مكتوب بلغة تفهمها أجهزة الكمبيوتر المادية. ويطلق على إنشاء هذه العمليات الحوسبية اسم البرمجة، ويطلق على اللغات التي تحدد البرامج اسم لغات البرمجة.

البرامج أدوات حدية

مفهوم البرنامج مفهوم محيِّر وغامض، بل إنه غريب. السبب الأول — كما سأوضح بعد برهة — هو إمكانية وصف العملية الحوسبية الواحدة على عدة مستويات من التجريد بناءً على اللغة التي تعبِّر عن تلك العملية، ما يتيح وجود عدة برامج «متكافئة». السبب الثاني هو أن البرنامج له وجهان مثل الإله يانوس؛ فالبرنامج — من جانب — «نصٌّ»

جامد، أي بنية رمزية تحتوي على كل سمات الأداة المجردة. ومن جانب آخر، تُعدُّ البرامج «عملية» ديناميكية؛ بمعنى أنها تتسبب في حدوث أشياء داخل جهاز الكمبيوتر المادي، وهذه العمليات تستهلك وقتًا ماديًّا ومساحة مادية؛ ومن ثَم فإن لها ركيزة مادية تعمل بناءً عليها. إضافة إلى ذلك، تكون البرامج بحاجة إلى وسيط مادي من أجل أن تعمل.

إذن، البرامج لها وجه مجرَّد وآخر مادي، ولذلك نقول إنها أدوات حدية. وتبعات هذه الحدية كبيرة ومثيرة للجدل على حد سواء.

أولًا، ينجذب البعض من أوساط علم الكمبيوتر إلى فكرة التجريد في البرامج ويرون أن البرامج كائناتٌ «رياضية». في نظرهم، البرمجة نوع من الأنشطة الرياضية التي تتضمن المسلَّمات والتعريفات والمبرهنات والبراهين. ويشدِّد علماء كمبيوتر آخرون على جانبها المادي ويرون أن البرامج كائناتٌ «تجريبية». ففي نظرهم، البرمجة نشاطٌ هندسي تجريبي يتضمن المهام الهندسية العادية، مثل تحديد المتطلبات والتصميم والتنفيذ وإجراء التجارب التي تختبر الأدوات الناتجة وتُقيِّمها.

ثانيًا، كثيرًا ما تشبَّه البرامج بالعقل. فإذا كان البرنامج كائنًا حدِّيًّا اصطناعيًّا، فالعقل كائن حدي طبيعي. فمن جهة، العمليات العقلية (أو المعرفية) مثل التذكُّر والتفكير والإدراك والتخطيط وفهْم اللغة وإتقانها وغيرها، هي عمليات يمكن دراستها (وقد جرت دراستها بالفعل على مدار قرون) وكأن العقل شيءٌ مجرد تمامًا يتفاعل مع العالم الواقعي تفاعلًا مستقلًّا. لكن العقل له مكان. وما لم يكن المرء من المتعصبين لنظرية ثنائية العقل والجسد الذين يفصِلون العقل عن الجسد بالكامل، فلن يصدق أن العقل يمكن أن يوجد خارج الدماغ — وهو شيء مادي. وبينما يدرُس بعض الفلاسفة وعلماء العلوم المعرفية العقل «وكأنه» كِيان مجرد، يجدُّ علماء الأعصاب في البحث عن تفسيرات مادية بحتة للظواهر العقلية من حيث العمليات التي تجري في الدماغ.

في واقع الأمر، تأثَّرت الدراسات العلمية لعملية المعرفة أو الإدراك تأثرًا كبيرًا بتشبيه العقل بالبرامج. ومن تبِعات هذا التأثر تطويرُ فرع من فروع علم الكمبيوتر يسمَّى «الذكاء الاصطناعي»، يحاول إنشاء أدوات حوسبية تشبه العقل وتشبه الدماغ. ومن التبِعات أيضًا تحويلُ علم النفس المعرفي إلى مجالٍ أشملَ يسمَّى «العلوم المعرفية»، من الفرضيات الجوهرية فيه أنه يمكن تصميمُ العمليات العقلية في صورة عمليات حوسبية على غرار البرامج.

ومن ثَم، فقد امتدَّ التأثير الفكري لعلم الكمبيوتر إلى ما هو أبعد من المجال نفسه. وبقدرِ ما توسَّعت نظرية داروين عن التطور إلى ما هو أبعد من علم الأحياء، تخطى تأثير علم الكمبيوتر إلى ما هو أبعد من عملية الحوسبة نفسها، وذلك بسبب تشبيه العقل بالبرامج. بل إن «فكرة» الحوسبة في ذاتها (كما سنرى في فصل لاحق) تجاوزت نطاقَ أجهزة الكمبيوتر المادية والحوسبة التلقائية. وأعتقد أن الإنصاف يقتضي القول بأن قلة من العلوم الاصطناعية فقط هي ما كان لها مثل هذه العواقب الفكرية خارج نطاقات مجالاتها.

من التبعات الأخرى لحدية البرامج، التشابك القوي بين البرامج والبرمجة وبين «اللغات الاصطناعية» التي تسمَّى لغات البرمجة. فبإمكان المرء أن يصمِّم خوارزميات باستخدام لغة طبيعية، وربما تكون معزَّزة ببعض الرموز الاصطناعية (مثلما رأينا في حالة الخوارزميات الواردة في الفصل الثالث). لكن لن يكون المرء مبرمجًا من دون أن يتقن لغة واحدة على الأقل من لغات البرمجة. يمكن أن تكون الصيغة البليغة (ولو كانت تقريبية) بالشكل التالي:

الخوارزميات + لغات البرمجة = البرامج

يترتَّب على هذه الصيغة في حد ذاتها عدة نتائج.

إحدى هذه النتائج نشوء نظرية لغات البرمجة باعتبارها فرعًا من فروع علم الكمبيوتر. وقد أدَّى هذا حتمًا إلى نشوء علاقة بين هذه النظرية وعلم اللغويات الذي يهتم بتراكيب اللغة الطبيعية.

نتيجة ثانية تتمثَّل في السعي الدءوب وراء ابتكار «أفضل» لغة برمجة، والتي يمكنها أن تنجز المطلوب من أي لغة على نحوٍ أفضل من أي لغة منافسة سابقة عليها أو معاصرة لها. هذا النشاط هو نشاط تصميم اللغة. والصعوبة التي تواجه مصمِّمو اللغة ذات بُعدين، وهما تسهيل توصيل العمليات الحوسبية لأجهزة الكمبيوتر المادية بحيث يمكنها إنجاز هذه العمليات بأقل حد من تدخُّل الإنسان؛ وكذلك تيسير التواصل مع البشر «الآخرين» بحيث يفهمون العملية الحوسبية ويحللونها وينقدونها ويقدمون اقتراحاتٍ لتحسينها تمامًا مثلما يفعل الناس مع أي نص. وقد كان هذا التحدي المزدوج مصدرَ هوس دائم لعلماء الكمبيوتر بالبرمجة ولغات الحوسبة الأخرى.

نتيجة ثالثة ترتبط بوضوح بتصميم اللغات وهي دراسة وتطوير البرامج التي تسمَّى البرامج المترجِمة التي تحوِّل البرامج المكتوبة بلغة برمجة إلى لغة الآلة الخاصة

بأجهزة كمبيوتر مادية بعينها. إلا أن تصميم البرامج المترجِمة وتنفيذها يعدُّ فرعًا آخر من فروع علم الكمبيوتر.

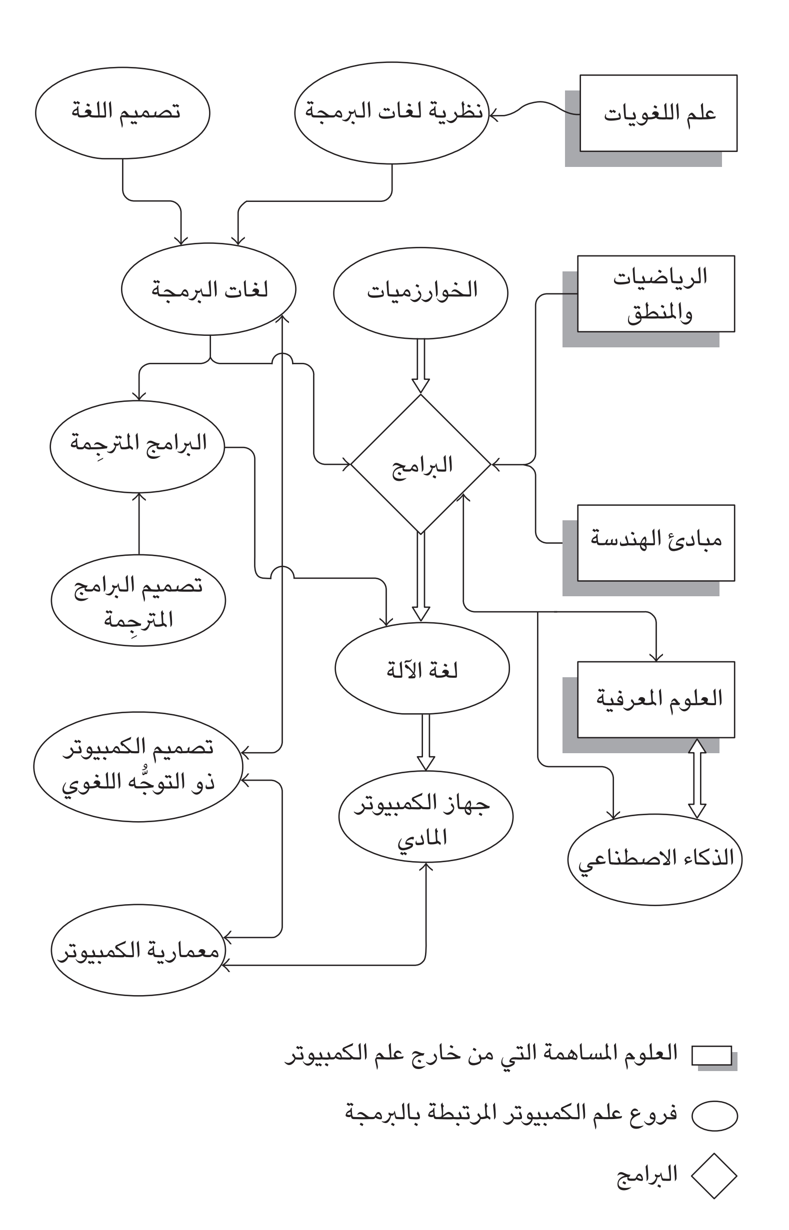
وأخيرًا، كانت ثمَّة جهود لتصميم سمات لأجهزة الكمبيوتر المادية من شأنها تسهيل مهمة البرنامج المترجِم. ويطلق على هذا النشاط «تصميم الكمبيوتر ذو التوجه اللغوي»، وقد حظي باهتمام كبير على مر التاريخ ضمن فرع من علم الكمبيوتر يطلق عليه معمارية الكمبيوتر (انظر الفصل الخامس).

ويوضح الشكل ٤-١ بطريقة تخطيطية العلاقات العديدة التي بين البرامج والبرمجة وبين تلك الكيانات والمجالات العلمية الأخرى. الكيانات المكتوبة داخل الأشكال المستطيلة مجالاتٌ مساهمة من خارج علم الكمبيوتر؛ والكيانات المكتوبة في الأشكال البيضاوية مجالاتٌ تندرج تحت علم الكمبيوتر.

اللغة والفكر والواقع والبرمجة

لاحظ أني قلت كلمة «لغة» في القسم السابق ولم أقل «ترميز». يشير الترميز إلى الرموز التي تُبتكر «للكتابة» عن شيءٍ ما. ومن الأمثلة على ذلك، الرموز الكيميائية أو الرياضية. أما اللغة فتتجاوز الترميز من حيث إنها توفِّر نسقًا من الرموز للتعبير عن «التفكير» بشأن شيءٍ ما. اللغة متداخلة مع الفكر ذاته.

ثمَّة قضية شهيرة أُجريت بشأنها سجالات بين اللغويين وعلماء الأنثروبولوجيا وهي: هل الفكر «تحدِّده» اللغة؟ يؤكد البعض ذلك قائلين إن اللغة التي نستخدمها تحدِّد الطريقة التي نفكِّر بها بشأن العالم والأشياء التي نفكر بها، بل إنها «تحدِّد» تصوُّرنا عن الواقع ذاته. من أكثر نتائج ذلك الرأي تطرفًا هو: بما أن اللغة عنصر محدد للثقافة، فلا يمكن ترجمة الأفكار أو التصورات أو المفاهيم من ثقافة لغة إلى ثقافة لغة أخرى. فكلٌّ منا محتجَز داخل الواقع الذي تحدِّده اللغة. إنها حالةُ ما بعد حداثية جدًّا. ويتبنى آخرون رأيًا أكثرَ اعتدالًا وهو أن اللغة «تؤثر» في طريقة تفكيرنا بشأن العالم ولكنها لا تحدِّدها. والاقتراح بأن اللغة تحدِّد الأفكار أو تؤثِّر فيها يسمى «فرضية سابير–وورف» نسبة إلى عالمَي علم اللغويات الأنثروبولوجية إدوارد سابير وبنجامين لي وورف اللذين صاغاها. (يطلق على هذه الفرضية أيضًا اسم «مبدأ النسبية اللغوية».)



شكل ٤-١: البرمجة وما يرتبط بها من فروع من داخل علم الكمبيوتر وعلوم من خارجه.

تناولت فرضية سابير–وورف اللغات الطبيعية. لكن اهتمامنا ينصب على اللغات الاصطناعية، وبالتحديد اللغات التي ابتُكرت للتعبير عن العمليات الحوسبية. وعلى حد علمي، لم يضَع أحد إطارًا لفرضية مماثلة لفرضية سابير–وورف في مجال الحوسبة، لكن الهوس الذي أظهره علماء الكمبيوتر منذ أوائل أيام أجهزة الكمبيوتر الإلكترونية تجاه البرمجة ولغات الحوسبة يشير بقوة إلى قبول بعض أشكال الفرضية قبولًا ضمنيًّا في أوساط علم الكمبيوتر. وبصورة أدق، يمكننا القول بقدرٍ من الثقة إن لغات الحوسبة (لا سيما لغات البرمجة) مرتبطة ارتباطًا وثيقًا بطبيعة بيئة الحوسبة؛ وأن لغات البرمجة تؤثِّر في عقلية المبرمجين.

لذا، لنتناول لغات البرمجة أولًا. أما طبيعة البرامج فستبرز بصورة طبيعية من هذه المناقشة.

لغات البرمجة كأدوات مجردة

ذكرنا من قبل أن عملية الحوسبة يمكن تحديدها باعتبارها برامجَ ذات مستويات تجريد مختلفة تقف على «مسافات» متباينة من أجهزة الكمبيوتر المادية التي تنفِّذ تلك البرامج. وفي المقابل، يمكن تصور لغات البرمجة على مستويات مختلفة من التجريد. هناك تقسيم بسيط يفرِّق بين اللغات «العالية المستوى» واللغات «المنخفضة المستوى». اللغات العالية المستوى تمكِّن من كتابة البرامج بمعزل عن أجهزة الكمبيوتر المادية التي ستنفِّذها، وتشير اللغات المنخفضة المستوى إلى اللغات المصمَّمة لفئات محددة من أجهزة الكمبيوتر أو حتى على نحوٍ أكثر تحديدًا لجهاز كمبيوتر بعينه.

من ثَم تكون اللغات العالية المستوى «مستقلة عن الآلة»، واللغات المنخفضة المستوى «معتمدة على الآلة»، مع التنبيه إلى أن درجة الاستقلال أو الاعتماد قد تتفاوت تفاوتًا كبيرًا. يطلق على اللغات ذات المستوى الأقل «لغات التجميع» وهذه اللغات مخصَّصة لأجهزة كمبيوتر مادية معينة (سواء لفئة منها أو لأجهزة فردية) لدرجةِ أن مبرمج لغة التجميع يتعامل حرفيًّا مع سماتِ أجهزة الكمبيوتر نفسها.

مرَّ تاريخ البرمجة بوقت كانت كل عمليات البرمجة تقريبًا تتم باستخدام لغات التجميع. حينذاك، كانت هذه البرامج لا تزال بنيات رمزية (ومجردة بدرجة محدودة للغاية)، ولكن كانت هناك برامج مترجِمة تسمى «برامج التجميع» تحوِّل هذه البنيات إلى لغة آلة لأجهزة الكمبيوتر المستهدفة. لكن بسبب الضجر والصعوبة ومقدار الوقت

الذي يستغرقه المبرمج واحتمالية الخطأ في البرمجة بلغة التجميع، تحوَّل التركيز إلى ابتكار وتصميم لغات برمجة ذات مستوًى أعلى بكثير ومستقلة عن الآلة، وأوكلت للبرامج المترجِمة مهمة ترجمة البرامج المكتوبة بهذه اللغات إلى لغات آلة مخصَّصة لأجهزة كمبيوتر بعينها.

في هذا الفصل وما بعده من فصول إلى نهاية الكتاب، يشير مصطلح «لغة البرمجة» إلى اللغات العالية المستوى ما لم يُذكر غير ذلك صراحةً.

وعلى خلاف اللغات الطبيعية، لغاتُ البرمجة لغاتٌ مبتكرة أو مُصمَّمة. ومن ثَم فهي أدوات. وتنطوي على استخدام الرموز. وكما سنرى، لغة البرمجة هي فعليًّا مجموعةٌ من البنيات الرمزية، وهي أدوات مجردة — كونها مستقلة عن أجهزة الكمبيوتر المادية — بالمعنى ذاته مثل الخوارزميات. ومن ثَم يكون لدينا وضعٌ مثير للتساؤل؛ ففي حين أن البرامج المكتوبة بهذه اللغات حدية، فإن لغات البرمجة ذاتها مجردة.

اللغة = ترميز + مفاهيم وفئات

لنراجع الفرْق بين الترميز واللغة. يزخر مجال الحوسبة بمئات اللغات الحوسبية. صُمِّمت غالبية هذه اللغات من أجل البرمجة، ولكن توجد لغات مخصَّصة لأغراض أخرى، لا سيما أغراض تصميم ووصف أجهزة الكمبيوتر المادية بمستويات التجريد المتنوعة (انظر الفصل الخامس). ويطلق عمومًا على هذه اللغات «لغات وصف عتاد الكمبيوتر» أو «لغات تصميم أجهزة الكمبيوتر ووصفها».

تستخدم اللغات الحوسبية هذه ترميزاتٍ — رموزًا — مختلفة، ويكمُن جزء من المجهود الذهني في تعلُّم لغة حوسبية جديدة في إتقان الترميز؛ أي، ما تشير إليه الرموز. وهذا ينطوي على الربط بين علامات الترميز و«مفاهيم» حوسبية و«فئات» لغوية أساسية. إذن تتألف اللغة من مجموعة مفاهيم وفئات، بالإضافة إلى الترميز الذي يمثِّلها. مرة أخرى، كصيغة تقريبية:

مفاهيم/فئات + ترميز = لغة

في الحوسبة، استخدمت علامات مختلفة في لغات مختلفة لترمز إلى مفهوم واحد. وعلى النقيض من ذلك، يمكن أن ترمز العلامة الواحدة إلى مفاهيمَ مختلفةٍ في لغات مختلفة.

على سبيل المثال، يمكن الإشارة إلى مفهوم البرمجة الجوهري المعروف باسم «التعيين» (والذي سبق أن ورد في الفصل الثالث) بعلامات مثل «=>»، «=»، «←»، «:=» في لغات مختلفة. ومن ثَم فإن عبارات التعيين التي بالشكل التالي:

$$X + 1 => X$$

$$X = X + 1$$

$$X := X + 1$$

$$X \longleftarrow X + 1$$

$$X += 1$$

كلها لها معنًى واحد وهو: القيمة الحالية للمتغير X تزيد بمقدار 1 وتُعيَّن النتيجة (أو تُنسخ مرة أخرى) إلى المتغير X. التعيين مفهومٌ حوسبي وعبارة التعيين هي فئة لغوية، كما أنها موجودة في معظم لغات البرمجة. وتختلف طرق تمثيلها من لغة إلى أخرى بناءً على ذوق مصممي اللغة وميولهم.

المفاهيم والفئات في لغات البرمجة

إذن ما هي تلك المفاهيم والفئات؟ تذكُر أن الحوسبة معالجةٌ للرموز؛ أو معالجةُ المعلومات بتعبيرٍ أكثر شيوعًا. «البداية هي المعلومات» — أو البيانات كما يحب أن يسميَها مصمِّمو اللغة — المراد معالجتها. ومن ثَم فإن المفهوم الأساسي المضمَّن في جميع لغات البرمجة هو ما نطلق عليه نوع البيانات. وكما ذكرنا في الفصل الأول، يحدِّد نوع البيانات طبيعةَ القيم التي يحتفظ بها عنصر البيانات (وإلا سمِّي متغيرًا)، بالإضافة إلى العمليات المسموح بأدائها على هذه القيم. أنواع البيانات إما «أولية» (أو أساسية) أو «مركَّبة» (أو مجمَّعة)، حيث إنها تتركب من أنواع بيانات أساسية أكثر.

في مثال المضروب المذكور في الفصل الثالث، لا يوجد سوى نوع البيانات الأولي «الأعداد الصحيحة غير السالبة»، وهذا يعني أن الأعداد الصحيحة الأكبر من أو تساوي صفرًا هي قيمٌ مقبولة للتعبير عن متغيرات هذا النوع، والعمليات الحسابية على الأعداد الصحيحة هي وحدها التي يمكن تنفيذها على متغيرات هذا النوع. وهذا يعني أيضًا أنه

لا يمكن تعيين غير القيم الصحيحة لمتغيرات هذا النوع. على سبيل المثال، عبارة تعيين مثل الآتية:

$$x \leftarrow x + 1$$

تكون عبارة مقبولة إذا كان المتغير x معرفًا بحيث يكون نوع بياناته الأعداد الصحيحة. وإذا لم يُعرف المتغير x على هذا النحو، أو إذا عُرف (مثلًا) بأنه سلسلة رموز (تمثل اسمًا ما)، فسيكون التعيين غير مقبول.

وليس بالضرورة أن يكون العدد نفسه عددًا صحيحًا ما لم يُعرف بهذا النحو. إذن من المنظور الحوسبي، رقم الهاتف ليس عددًا صحيحًا؛ بل هو سلسلة رموز رقمية؛ فليس بإمكان المرء جمْع رقمَي هاتفين أو ضربهما. ومن ثَم إذا عُرف المتغير x بأنه سلسلةُ رموز رقمية، فإن عبارة التعيين السابقة لن تكون صالحة.

تحتوي خوارزمية البحث الخطي المذكورة في الفصل الثالث على أنواع بيانات أولية ومركَّبة. فالمتغير i من النوع الأوَّلي «الأعداد الصحيحة»، أما المتغير *given* فمن النوع المركَّب «سلسلة الرموز»، الذي هو ذاته مؤلَّف من النوع الأولي «الرموز». وقائمة الطلاب *student* أيضًا ذات نوع بيانات مركَّب، ويطلق عليها في بعض الأحيان «قائمة خطية» وفي أحيانٍ أخرى «مصفوفة». والعنصر الذي ترتيبه i من القائمة *student* له أيضًا نوع بيانات مركَّب (والذي يُطلق عليه أسماء مختلفة في لغات البرمجة المختلفة، منها «السجل» و«الصف») ويتألف هنا من نوعَي بيانات؛ الأول (*name*) من نوع سلسلة الرموز، والآخر (*email*) أيضًا من نوع سلسلة الرموز. إذن متغير مثل *student* عبارة عن «بنية بيانات» مرتَّبة هرميًّا؛ أي، رموز منظَّمة في شكل سلاسل رموز، وسلاسل رموز منظمة في شكل صفوف أو سجلات، وصفوف منظمة في شكل قوائم أو مصفوفات.

لكن البيانات أو المعلومات ليست سوى بداية أي عملية حوسبة. إضافة إلى ذلك، المتغيرات نفسها سلبية. فالحوسبة تتضمن إجراءات، وتضمين هذه الإجراءات في عمليات. ومن ثَم يجب ألا تتضمنَ لغة البرمجة وسيلةً لتحديد عناصر البيانات فحسب، بل يجب أن تتضمن أيضًا «عبارات» تحدِّد الإجراءات والعمليات.

في الحقيقة، صادفنا بالفعل ولعدة مرات «أنواع» العبارات الأكثر جوهرية في الفصل السابق. أحد هذه الأنواع هو عبارة التعيين التي يستدعي تنفيذها عمليةً تتضمن كلًّا من الاتجاه والتدفق الزمني للمعلومات. على سبيل المثال، في تنفيذ عبارة التعيين:

$$A \leftarrow B$$

حيث A وB عبارة عن متغيرين، تتدفَّق المعلومات من B إلى A. ولكنه ليس كتدفُّق المياه من وعاء إلى آخر. فقيمة B لا تتغير ولا تقل ولا تفرُغ بعد تنفيذ هذه العبارة. بل إن قيمة B «تُقرأ» و«تُنسخ» إلى A بحيث تتساوى قيمة A مع قيمة B في النهاية. لكن في تنفيذ العبارة:

$$\mathrm{A} \longleftarrow \mathrm{A} + \mathrm{B}$$

تتغير قيمة A بالفعل، حيث إن قيمة A الجديدة تساوي قيمة A القديمة مضافًا إليها قيمة B. لكن قيمة B تبقى من دون تغيير.

الصيغة العامة لعبارة التعيين هي:

$$\mathrm{X} \longleftarrow \mathrm{E}$$

حيث E «تعبيرٌ» (مثل التعبيرات الرياضية $Y + 1$، و$((X - Y) * (Z/W))$). وبوجه عام، تنفيذ هذه العبارة يكون بعملية مكوَّنة من خطوتين: أولًا، يجري تقدير E؛ ثم تعيَّن هذه القيمة إلى X.

إذن تحدد عبارة التعيين وحدةَ الإجراء في العملية الحوسبية. إنها بمثابة العملية الأولية في العمليات الحوسبية. لكن مثلما تلتحم الذرات في العالم الطبيعي لتكوِّن الجزيئات وتلتحم الجزيئات لتكوِّن جزيئاتٍ أكبر، تحدث عملية مشابهة في عالم الحوسبة. إذ تلتحم عبارات التعيين لتكوين مقاطعَ أكبر، وتتَّحد المقاطع لتكوين مقاطع أكبر إلى أن نحصل على برامج كاملة. ثمة تسلسل هرمي موجود في العالم الحوسبي، كما هو موجود في الطبيعة.

من ثَم كانت أكبر مهمة لدى علماء الكمبيوتر هي اكتشاف «قواعد التركيب»، وابتكار أنواع عبارات تمثِّل هذه القواعد، وتصميم ترميزات لكل نوع من أنواع العبارات. وفي حين أن قواعد التركيب وأنواع العبارات يمكن أن تكون عامة إلى حد كبير، يمكن للغات البرمجة المختلفة أن تستخدم ترميزات مختلفة لتمثيلها.

من أنواع العبارات العبارة «التسلسلية»؛ وتعني تركيب عبارتين أو أكثر (أبسط) تركيبًا تسلسليًّا بحيث يتم تنفيذهما حسب ترتيب العبارات المكوَّنة. الرمز الذي استخدمته في الفصل الثالث للإشارة إلى التسلسل هو الفاصلة المنقوطة «;». ومن ثَم، نجد العبارة

التسلسلية التالية في خوارزمية إقليدس:

$$m \leftarrow n;$$

$$n \leftarrow r;$$

goto step 1

وفيها يتقدَّم تدفُّق التحكم عبْر العبارات الثلاث حسب الترتيب الموضَّح.

لكن قد تتطلب العمليات الحوسبية أيضًا الاختيار من بين عدة بدائل. وعبارات if…then…else التي استخدمناها في الفصل الثالث في عدة خوارزميات هي أمثلة على نوع العبارات «الشرطية» في لغات البرمجة. في الصيغة العامة if C $then$ $S1$ $else$ $S2$ يتم تقييم الشرط C وإذا كان صحيحًا، ينتقل التحكم إلى $S1$، وإلا تدفَّق التحكم إلى $S2$.

ونحتاج في بعض الأحيان إلى إعادة تدفُّق التحكم إلى جزء سابق من العملية الحوسبية وتكراره. ويُعَد الترميزان while…do وrepeat…until في خوارزمية البحث الخطي وخوارزميات المضروب غير ذاتية الاستدعاء في الفصل الثالث كأمثلة على نوع عبارات «التكرار».

أنواع العبارات الثلاث هذه — التسلسلية والشرطية والتكرارية — أسسٌ لإنشاء البرامج. وتوفِّر كل لغة برمجة ترميزًا لتمثيل هذه الفئات. في الحقيقة ومن حيث المبدأ، يمكن تحديد أي عملية حوسبية ببرنامج يحتوي على مزيج من أنواع العبارات الثلاث هذه. (في عام ١٩٦٦، هذا الأمر أثبته مُنظِّرا علم الكمبيوتر الإيطاليان كورادو بوم وجوسيبي جاكوبيني.) ومن المنظور العملي، طُرح العديد من قواعد التركيب وأنواع العبارات المقابلة لتسهيل البرمجة (مثل «التفريع غير الشرطي» الذي تمثِّله عبارة goto المستخدَمة في خوارزمية إقليدس الواردة بالفصل الثالث).

البرمجة باعتبارها فنًّا

البرمجة عمليةُ تصميم، ومثل كل أنشطة التصميم، تنطوي على حسن التقدير والحَدْس والذوق الجمالي والخبرة. ولهذا السبب وضع دونالد كنوث العنوان «فن برمجة الكمبيوتر» لسلسلة أفروداته الشهيرة والمؤثِّرة (التي ظهرت فيما بين عامي ١٩٦٨ و١٩٦٩). وبعد عَقد تقريبًا، أوضح كنوث رأيه في هذا الموضوع في إحدى محاضراته. كتب أنه عندما

يأتي الحديث عن فن البرمجة، فإنه يشير إلى البرمجة على أنها «ضربٌ من ضروب الفن». ينبغي أن تُشبع البرامج الذوقَ الجمالي؛ إذ ينبغي أن تكون جميلة. وتجربة كتابة برنامج ينبغي أن تشبه نَظم قصيدة أو تلحين مقطوعة موسيقية. ومن ثَم، فإن فكرة «الأسلوب» — وهي جزء وثيق الصلة بسياقات الفن والموسيقى والأدب — يجب أن تكون عنصرًا من مظاهر الجمال في البرمجة. ولنتذكر مقولةَ عالِم الكمبيوتر الهولندي إدسخر ديكسترا المذكورة في الفصل الثالث: في ابتكار الخوارزميات، «الجمال هو ما نسعى إليه». وقد عبَّر عالِم علم الكمبيوتر الروسي إيه بي يرشوف عن وجهة نظر مشابهة.

وفي إطار هذا الموضوع، اقترح كنوث في وقتٍ لاحق أنه ينبغي بالبرامج أن تندرج ضمن «الأعمال الأدبية»؛ بمعنى أن يستمتع المبرمج بكتابة البرامج وأن تمنح هذه البرامج البهجةَ لدى قراءة الآخرين لها. (أطلق كنوث على هذه الفلسفة اسم «البرمجة المتعلمة»، وإن كنتُ أعتقد أن «البرمجة الأدبية» ستكون أليقَ للتعبير عن رأيه.)

البرمجة كعلم رياضي

بالطبع يسعى علماء الكمبيوتر (بمن فيهم كنوث) إلى اكتشافِ أسسٍ صورية وموضوعية أكثر للبرمجة. فهم يبتغون علمًا للبرمجة. والنتيجة التي توصَّل إليها كلٌّ من بوم وجاكوبيني والتي سبق ذكرها هي نوع من النتائج الرياضية الصورية التي يتوق إليها علماء الكمبيوتر. وفي الحقيقة، يرى كثير من علماء الكمبيوتر أن «علم» البرمجة علمٌ «رياضي».

رؤية البرمجة باعتبارها علمًا رياضيًّا تجلَّت بشكل بارز بثلاث طرق أخرى تتعلق جميعها بالوجه المجرَّد للبرامج أو — كما أشرت مسبقًا في هذا الفصل — بوجهة النظر التي يتبناها البعض بأن البرامج كائناتٌ رياضية.

المساهمة الأولى في علم البرمجة هي اكتشاف «قواعد التركيب اللغوي» في لغات البرمجة. تلك قواعد تحدِّد السلامة النحوية في البرامج ولها تأثير عملي ضخم، حيث إن من أوائل مهام البرامج المترجِمة (التي تحوِّل البرامج العالية المستوى تلقائيًّا إلى لغة آلة) هي التأكد من سلامة النحو أو التركيب في البرامج التي تترجمها. وترجع بدايات ظهور نظرية التركيب اللغوي في لغات البرمجة إلى جهودِ عالِم اللغويات نعوم تشومسكي في نظرية التركيب اللغوي (في اللغات الطبيعية).

المساهمة الثانية في علم البرمجة هي تطوير «قواعد الدلالة»؛ وهي المبادئ التي تحدِّد معنى أنواع العبارات المختلفة. ويجب أن تكون أهميتها واضحة؛ فمن أجل استخدام لغة

البرمجة لا بد أن يكون المبرمج على وعي تامٍّ بمعاني أنواع العبارات المكوِّنة لهذه اللغة. وكذلك، يجب أن يفهم كاتب البرنامج المترجِم ومن دون لبسٍ معنَى كل نوع من أنواع العبارات من أجل أن يترجم البرامج إلى لغة آلة. ولكن الدلالة، حسب معناها المستخدَم في علم اللغويات، تُعَد مشكلة شائكة حيث إنها تتضمن ربط فئات لغوية بما تشير إليه في العالم، ونظرية الدلالة في لغات البرمجة تعكس هذه الصعوبات ذاتها. ومن المناسب أن نقول إن نظرية الدلالة في البرمجة — على الرغم من صياغتها المتطورة — لم تلقَ القبول ذاته في أوساط علم الكمبيوتر ولم تُستخدم بقدْر الفاعلية ذاته مثل نظرية التركيب اللغوي.

المساهمة الثالثة في علم البرمجة ترتبط ارتباطًا وثيقًا بقضية الدلالة. قامت هذه المساهمة على قناعة علماء كمبيوتر أمثال الإنجليزي سي إيه آر هور والهولندي إدسخر ديكسترا بأن الحوسبة شبيهة بالرياضيات، وأنه يمكن تطبيق المبادئ ذاتها التي يستخدمها علماء الرياضيات — مثل المسلَّمات وقواعد المنطق الاستنتاجي والمبرهنات والبراهين — على البرمجة. وقد صاغ هور هذه الفلسفة صياغةً صريحة وجريئة؛ حيث أعلن عام ١٩٨٥ البيان التالي:

(أ) **أجهزة الكمبيوتر آلات رياضية.** بمعنى أنه يمكن تحديد سلوكها رياضيًّا ويمكن اشتقاق كل تفصيلة من التعريف اشتقاقًا منطقيًّا.

(ب) **البرامج تعبيرات رياضية.** فهي تصف بدقة وبالتفصيل سلوك الكمبيوتر الذي تنفَّذ عليه.

(جـ) **لغة البرمجة نظرية رياضية.** فهي نظام صوري يساعد المبرمج في كل من تطوير البرنامج، وكذلك إثبات أن البرنامج يستوفي مواصفات متطلباته.

(د) **البرمجة نشاط رياضي.** ممارسة البرمجة تتطلب تطبيقَ الطرق التقليدية في الفهْم والإثبات الرياضي.

بحسب التقليد المسلماتي الشهير في الرياضيات، يبدأ المرء بالمسلَّمات (وهي الافتراضات التي تعتبر صحيحة «بديهيًّا» بشأن المجال المعني، مثل مبدأ الاستقراء الرياضي المذكور في الفصل الثالث)، وتعريفات المفاهيم الأساسية، ثم «يقيم الدليل» تدريجيًّا على رؤًى واقتراحات جديدة (يطلق عليها مجتمعة المبرهنات) مشتقة من هذه المسلَّمات والتعريفات والمبرهنات التي ثبتت صحتها بالفعل، وذلك باستخدام قواعد

الاستنتاج. وانطلاقًا من هذا التقليد، تُعنى المساهمة الثالثة في علم البرمجة الرياضي بصياغةِ براهينَ مسلماتية بشأن صحة البرامج بناءً على المسلَّمات والتعريفات وقواعد الاستنتاج التي تحدِّد الدلالة في لغة البرمجة ذات الصلة. يطلق على الدلالة «الدلالة المسلَّماتية» ويُعرف تطبيقها باسم البراهين المسلَّماتية الخاصة بالصحة.

ومثلما هو الحال مع المنهج المسلَّماتي في الرياضيات (واستخدامه في مجالات مثل الفيزياء الرياضية والاقتصاد)، يزخر علم البرمجة الرياضي بمظاهر الجمال والتميز الصوري. لكن حري بنا أن نشير إلى أنه على الرغم من تطوير كمٍّ هائل من المعرفة في هذا المجال، فإنه يظل عدد لا بأس به من علماء الكمبيوتر الأكاديميين والممارسين في المجال متشككين بشأن قابلية تطبيقها بشكل عملي في «العالم الواقعي» المضطرب للحوسبة.

البرمجة كهندسة (برمجيات)

البرمجة ضربٌ من ضروب الهندسة نظرًا لأن الكثيرين يرون أن البرامج ليست مجرد أدوات جميلة مجردة وحسب. فحتى هور في بيانه يقرُّ بأن البرامج يجب أن تصف سلوك «أجهزة الكمبيوتر» التي تنفِّذها. فأجهزة الكمبيوتر هي الوجه المادي للبرامج وهي التي تضفي عليها صفةَ الحدية. وفي نظر الكثيرين في واقع الأمر، البرامج منتجاتٌ تكنولوجية، ومن ثَم تصبح البرمجة نشاطًا هندسيًّا.

يبدو أن كلمة «برمجيات» دخلت إلى مفردات الحوسبة عام ١٩٦٠. لكن تبقى دلالاتها غيرَ محدَّدة. فالبعض يستخدمون كلمتي «برمجيات» و«برنامج» على أنهما مترادفتان. ويعتقد البعض أن البرمجيات تعني مجموعةَ البرامج الخاصة والضرورية (مثل أنظمة التشغيل والأدوات و«الأدوات المساعدة» الأخرى التي يطلق عليها مجتمعة «برامج النظام») المصمَّمة لتعمل على كمبيوتر مادي لإنشاء أجهزة افتراضية (أو أنظمة كمبيوتر) يمكن للآخرين استخدامها بمزيد من الكفاءة (ارجع إلى الشكل ٢-١ في الفصل الثاني). لكن لا يزال آخرون يعتقدون أن البرمجيات لا تعني البرامج فحسب، بل معها التوثيق ذو الصلة الذي هو ضروري من أجل تطوير البرامج الكبيرة وتشغيلها وصيانتها وتعديلها. وهناك البعض ممن يدرجون في هذا الإطار المعرفة والخبرة البشرية.

على أية حال، «البرمجيات» لها الدلالات المهمة التالية: إنها ذلك الجزء من نظام الكمبيوتر الذي لا يعتبر ماديًّا؛ وهي تتطلب وجودَ الكمبيوتر المادي لتشغيلها؛ وهناك ما يدُل على أنها منتج «صناعي» إلى حد كبير، بكلِّ ما تدل عليه الصفة.

إذن، فالبرمجيات أداةٌ حوسبية تسهِّل على العديد من المستخدمين (ربما الملايين أو المليارات) استخدامَ أنظمة الكمبيوتر. وفي أغلب الأحيان (وإن لم يكن على الدوام) تكون البرمجيات أداةً منتجة تجاريًّا تُظهر مستويات معينة من الدقة والموثوقية التي أصبحنا نتوقَّعها من الأنظمة الصناعية.

ربما بالتناظر مع الأنظمة الصناعية الأخرى، يرتبط مشروع تطوير البرمجيات بـ «دورة حياة». ومن ثَم وكما هو الحال مع العديد من المشروعات الهندسية المعقَّدة الأخرى (مثل مشروع إطلاق قمر صناعي جديد في الفضاء)، يعتبر تطوير نظام برمجي مشروعًا هندسيًّا، وفي هذا السياق، يبدو أن مصطلح «هندسة البرمجيات» (الذي صيغ للمرة الأولى في أواسط ستينيات القرن العشرين) ملائمًا على نحوٍ خاص. وقد أدَّى هذا بطبيعة الحال إلى فكرة «مهندس برمجيات». فليس من قبيل الصدفة أن قدْرًا كبيرًا من التفكير بشأن هندسة البرمجيات منشؤه القطاع الصناعي.

وقد طُرحت نماذج مختلفة لدورة حياة البرمجيات على مدار الخمسين عامًا الماضية. وبوجه عام، تقرُّ جميعها بأن تطوير نظام البرمجيات يتضمن عددًا من المراحل:

(أ) تحليل المتطلبات التي تهدُف البرمجيات إلى استيفائها.

(ب) تحديد المواصفات الدقيقة للوظائف والأداء والتكلفة لمختلف المكونات («الوحدات») التي يمكن تحديدها من تحليل المتطلبات.

(ج) تصميم نظام البرمجيات الذي (يُرجى) أن يلبي المواصفات. وهذا النشاط قد يتكوَّن في ذاته من مراحل تصميم مفاهيمية ومفصلة.

(د) تنفيذ التصميم باعتباره نظامَ برمجيات تشغيليًّا محددًا بلغة برمجة وتجميعه للتنفيذ على نظام (أنظمة) الكمبيوتر المستهدف.

(هـ) التحقق والتثبُّت من مجموعة البرامج المنفَّذة لضمان أنها تستوفي المواصفات.

(و) بمجرد إتمام عمليتي التحقُّق والتثبُّت، تأتي صيانة النظام، وتعديله إذا لزم الأمر ووقتما يلزم.

بالطبع لا تتوالى هذه الخطوات بطريقة خطية صارمة. فدائمًا هناك احتمال الرجوع إلى مرحلة سابقة من مرحلة متقدمة إذا اكتُشفت عيوب وأخطاء. إضافة إلى ذلك، تتطلب دورة حياة البرمجيات أيضًا بنيةً أساسية — من الأدوات والمنهجيات ومعايير التوثيق والخبرة البشرية، وهذه العناصر مجتمعة تكوِّن «بيئة» هندسة البرمجيات.

كذلك لا بد من ملاحظة أن مرحلة أو أكثر من هذه المراحل ستستلزم نظريات علمية محكمة باعتبارها جزءًا من تنفيذها. وربما تنطوي المواصفات والتصميم على استخدام لغات لها قواعد التركيب والدلالة الخاصين بها؛ كما أن التصميم والتنفيذ المفصَّلين سينطويان على لغات برمجة، وربما يتضمنان استخدام أساليب البرهان المسلَّماتية. كذلك لا مناص من أن التحقُّق والتثبُّت سيتطلبان أنماطًا متطورة من الإثبات والاختبار التجريبي للبرمجيات. وبقدرِ ما تستلزم المجالات الكلاسيكية للهندسة (مثل الهندسة الإنشائية أو الميكانيكية) العلوم الهندسية باعتبارها مكونات لها، فكذلك الأمر مع هندسة البرمجيات.

الفصل الخامس

مجال معمارية الكمبيوتر

يقع الكمبيوتر المادي في نهاية التسلسل الهرمي «جهاز الكمبيوتر» الموضَّح في الشكل ٢-١. وفي اللغة الدارجة، يشار إلى جهاز الكمبيوتر المادي باسم «العتاد» (hardware). وكلمة hard في المقابل الإنجليزي للكلمة تعني أنه أداةٌ مادية تخضع لقوانين الطبيعة في نهاية الأمر. وجهاز الكمبيوتر المادي هو الأداة الحوسبية «المادية» الأساسية التي تهم علماء الكمبيوتر.

لكن إذا طرح أحدٌ السؤالَ التالي: «ما طبيعة جهاز الكمبيوتر المادي؟» فربما أراوغ في إجابتي. يرجع السبب إلى أن جهاز الكمبيوتر المادي، وإن كان جزءًا من تسلسل هرمي أكبر، فإنه معقَّد لدرجةِ أنه يحوي تسلسله الهرمي الداخلي الخاص به. ومن ثَم يمكن تصميمه ووصفه على عدة مستويات من التجريد. والعلاقة بين هذه المستويات تجمع بين مبادئ التسلسل الهرمي التركيبي، التجريدي/التفصيلي، والبنائي التي أوضحناها في الفصل الأول.

ربما أهمُّ جانب في هذا التسلسل الهرمي من وجهة نظر عالِم الكمبيوتر (ويرجع الفضل في جزء كبير من هذا الرأي إلى عالِم الرياضيات والقامة العلمية الأمريكي المجري جون فون نيومان؛ حيث إنه أول مَن توصَّل إليه) هو التفريق بين الكمبيوتر المادي باعتباره أداةً حوسبية تعالج الرموز والمكونات المادية التي تتبع قوانين الفيزياء والتي تتكون منها هذه الأداة. هذا التفريق مهم. فباعتبار الكمبيوتر أداة تعالج الرموز، يصبح أداة مجردة بنفس المعنى الذي ينطبق على البرمجيات، ولكن كما هو الحال مع البرمجيات، فهذه الأداة المجردة لا يكون لها وجود من دون تنفيذها ماديًّا.

رؤية الكمبيوتر المادي باعتباره أداةً حوسبية مجردة تعالج الرموز تشكِّل «معمارية» الكمبيوتر. (استُخدم مصطلح «المعمارية» من قبلُ للإشارة إلى الهيكل الوظيفي للأجهزة

الافتراضية الموضَّحة في الشكل ٢-١. لكن، الآن، مصطلح «معمارية الكمبيوتر» له دلالة فنية محدَّدة أكثر.) فالمكونات المادية (الرقمية) التي تنفِّذ معماريته — العتاد الفعلي — تشكِّل «تكنولوجيا» الكمبيوتر. ومن ثَم يتضح لنا الفرق بين «معمارية الكمبيوتر» والتكنولوجيا (الرقمية).

ثمَّة جانب آخر مهم في هذا التفريق. يكمُن هذا الجانب في إمكانية تنفيذ معمارية بعينها باستخدام تكنولوجيات مختلفة. فالمعمارية ليست مستقلة عن التكنولوجيات؛ حيث إن تطور التكنولوجيا يؤثِّر في التصميم المعماري، لكن هناك قدْر معيَّن من الاستقلالية أو «درجات من الحرية» التي يتمتع بها مصمِّم معماريات الكمبيوتر. وفي المقابل، يمكن لتصميم معماريةٍ ما أن يشكِّل نوع التكنولوجيا التي يتم استخدامها.

لتوضيح الصورة، لنضرب مثلًا بمؤسسة ولتكن جامعة. هذه المؤسسة لها من السمات المجردة والمادية. فتنظيم الجامعة ووحداتها الإدارية والأكاديمية المتعددة وبنيتها ووظائفها الداخلية وغير ذلك هو نظير معمارية الكمبيوتر. فبإمكان المرء أن يصمِّم جامعة (وهي أداة في نهاية الأمر) ويصفها ويناقشها ويحللها وينقدها ويغيِّر هيكلها تمامًا مثلما يمكن له أن يفعل مع أي كيان مجرد آخر. لكن الجامعة تُبنى بموارد بشرية ومادية. وهذه هي نظير تكنولوجيا (عتاد) الكمبيوتر. ومن ثَم في حين ينطوي تصميم الجامعة أو تطورها على قدْر كبير من الاستقلالية، يمكن أن يعتمد إنشاؤها فقط على طبيعة مواردها ووفرتها وكفاءتها (الموظفون والمباني والمعدَّات والمساحة المادية وبنية الحرم ككل وغير ذلك). وفي المقابل، إن تصميم تنظيم الجامعة سيؤثِّر في أنواع الموارد التي ينبغي توافرها.

وفيما يلي توضيح المصطلحات الأساسية المتعلقة بهذا النقاش: «معمارية الكمبيوتر» هي أحد فروع علم الكمبيوتر وتهتم بتصميم الكمبيوتر المادي ووصفه وتحليله ودراسة تنظيمه المنطقي وسلوكه وعناصره الوظيفية؛ وكل هذا يشكِّل معمارية الكمبيوتر (المادي). مهمة اختصاصي معمارية الكمبيوتر هي تصميم معماريات تلبي احتياجات مستخدمي الكمبيوتر المادي (مهندسو البرمجيات والمبرمجون ومصمِّمو الخوارزميات والمستخدمون غير الفنيين) من جانب، ومن جانب آخر تكون صالحة من الناحيتين الاقتصادية والتكنولوجية.

بذلك نجد أن معماريات الكمبيوتر أدواتٌ حدية. ولا بد لاختصاصي معمارية الكمبيوتر أن يكون ماهرًا في الجمع بين المتطلبات الوظيفية والأدائية للكمبيوتر، والجدوى التكنولوجية.

أجهزة الكمبيوتر الأنوية وذات النزعة الاجتماعية

يرجع قدْر كبير من الفضل في عولمة كل شيء إلى الكمبيوتر. إذا كان «الإنسان ليس جزيرة منعزلة»، فليس الكمبيوتر كذلك في القرن الحادي والعشرين. لكن فيما مضى وعلى مدار سنوات عديدة، كانت أجهزة الكمبيوتر بمثابة جزر منعزلة بالفعل. فالأداة الحوسبية من النوع الموضَّح باسم «النص» في الشكل ٢-١ كانت ستنجز المهام المنوطة بها وكأنه لا شيء في العالم سواها. كذلك ما كان لها تفاعل مع البيئة إلا بالبيانات والأوامر المُدخلة والنتائج المُخرجة. وخلاف ذلك، وتحقيقًا لكل الأغراض العملية، فإن الكمبيوتر المادي — ونظامه المخصَّص له وبرامج التطبيقات والأدوات الأخرى الخاصة به (مثل لغات البرمجة) — عاشت في عزلة أنوية بديعة.

لكن كما أوضحنا لتوِّنا، تضاءلت أعداد أجهزة الكمبيوتر الأنوية في الوقت الراهن. فابتكار الإنترنت وإنشاء البريد الإلكتروني وشبكة الويب العالمية ومختلف أشكال وسائل التواصل الاجتماعي وضعت حدًّا للأنوية الحوسبية. لكن حتى أكثر الأدوات خصوصية للمستخدم مثل الكمبيوتر المحمول أو الهاتف الذكي تصبح اجتماعية بمجرد أن يتصل المستخدم بالإنترنت ليشتريَ كتابًا أو يطلع على حالة الطقس أو يبحث عن الاتجاهات كي يذهب إلى مكانٍ ما. يصبح كمبيوتر المستخدِم اجتماعيًّا بمعنى أنه يتفاعل ويتواصل مع عددٍ لا حصر له من أجهزة الكمبيوتر الأخرى (على الرغم من أنه لا يعرفها) المنتشرة ماديًّا في ربوع الكوكب عبْر شبكة تسمَّى الإنترنت. في الحقيقة، كل مستخدم يرسل بريدًا إلكترونيًّا أو يبحث عن معلومة أو يشاهد مقطع فيديو عبْر الإنترنت ليس مجرد مستخدم لشبكة الإنترنت، بل يكون جهاز الكمبيوتر الخاص به جزءًا من شبكة الإنترنت. بل إن شبكة الإنترنت مجتمعٌ عالمي تفاعلي قوامه الأدوات الحوسبية ذات النزعة الاجتماعية والكيانات البشرية.

لكن توجد شبكات أبسط يمكن أن تكون أجهزةُ الكمبيوتر جزءًا منها. فالأجهزة داخل مؤسسة (مثل جامعة أو شركة) تتصل بعضها ببعض عبْر ما يسمَّى «الشبكات المحلية». كذلك يمكن لمجموعة من أجهزة الكمبيوتر الموزَّعة عبْر منطقةٍ ما أن تتعاون بعضها مع بعض، ويؤدي كل جهاز منها المهمةَ الحوسبية المنوطة به، ولكنها تتبادل المعلومات بعضها مع بعض عند الحاجة. وقد جرت العادة على تسمية هذه الأنظمة باسم أنظمة «الحوسبة المتعددة» أو «الحوسبة الموزَّعة».

تتم إدارة الحوسبة المتعدِّدة أو الحوسبة الموزَّعة أو الحوسبة عبْر الإنترنت بموجب مجموعة من قواعد عمل الشبكات تسمى «البروتوكولات» وأنظمة برمجيات بالغة التعقيد.

لكن عندما ننظر إلى مجال معمارية الكمبيوتر، فإن ما يهمنا هو الكمبيوتر بمفرده، سواء كان أنويًّا أو اجتماعي النزعة. وهذا ما سنتناوله فيما تبقى من هذا الفصل.

المعماريات الخارجية والداخلية

دخلت كلمة «معمارية» في سياق أجهزة الكمبيوتر للمرة الأولى في أوائل ستينيات القرن العشرين على يد ثلاثة من مهندسي شركة آي بي إم، وهم جين أمدال وفريدريك بروكس وجيريت بلاو. استخدم ثلاثتهم هذا المصطلح للإشارة إلى مجموعة السمات الوظيفية للكمبيوتر المادي كما هو متاح لمبرمج الكمبيوتر ذي المستوى الأدنى (مبرمجي الأنظمة الذين يبنون أنظمة التشغيل والبرامج المترجِمة والأدوات المساعدة الأساسية الأخرى التي تستخدم لغات التجميع)؛ أو «واجهتها الخارجية» إن جاز التعبير. لكن منذ ذلك الحين، امتدت ممارسة معمارية الكمبيوتر كي تشملَ التنظيم الداخلي المنطقي والهيكلي والوظيفي، وكذلك سلوك المكونات المادية للجهاز. لذا من المنظور العملي، يشير مصطلح «معمارية الكمبيوتر» إلى الجوانب الوظيفية والمنطقية لكلٍّ من الواجهة الخارجية والمكونات الداخلية للكمبيوتر المادي. لكن لم يتم الاتفاق على مصطلحين يشيران إلى هذين الجانبين، وانطلاقًا من مبدأ التبسيط، سأطلق عليهما المعمارية الخارجية والمعمارية الداخلية، على التوالي.

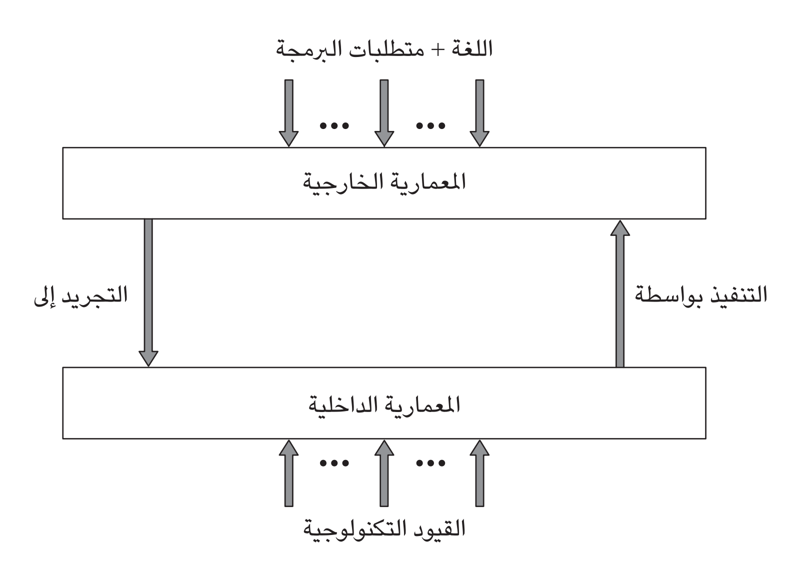
يرتبط هذان الجانبان بعلاقة تسلسل هرمي. فهما جانبان مجرَّدان مختلفان للكمبيوتر المادي، حيث تكون المعمارية الخارجية فيه شكلًا مجردًا للمعمارية الداخلية، أو، بالعكس، تكون المعمارية الداخلية تفصيلًا للمعمارية الخارجية. أو بناءً على استراتيجية التصميم المستخدَمة، يمكن اعتبار معمارية الكمبيوتر الداخلية تنفيذًا للمعمارية الخارجية.

ويتشكَّل تصميم معماريات الكمبيوتر الخارجية حسب القوى المبذولة من بيئة الكمبيوتر الحوسبية؛ أي بما أن المعمارية الخارجية هي الواجهة بين الكمبيوتر المادي ومبرمجي الأنظمة الذين يصمِّمون الأجهزة الافتراضية التي «يراها» المستخدم «العادي» للكمبيوتر، فمن الطبيعي أن يكون للمتطلبات الوظيفية التي تفرضها هذه البيئة تأثيرٌ في تصميم المعمارية الخارجية. على سبيل المثال، إذا كان الهدف من الكمبيوتر C دعم التنفيذ الفعَّال للبرامج المكتوبة بنوع معين من اللغات ولنقل L، فإن المعمارية الخارجية للكمبيوتر C قد تكون موجَّهة صوب سمات اللغة L؛ ومن ثَم تسهِّل مهمة البرنامج المترجِم

في ترجمة البرامج المكتوبة باللغة L إلى لغة آلة للكمبيوتر C. أو إذا كان نظام التشغيل OS الذي يحتل مستوًى أعلى من الكمبيوتر C يتضمن إمكانيات معينة، فيمكن تسهيل تنفيذ نظام التشغيل OS عبر سمات مناسبة تُدمج في المعمارية الخارجية للكمبيوتر C.

على الجانب الآخر، بما أن المعمارية الداخلية للكمبيوتر ستنفَّذ باستخدام مكونات (عتاد) مادية، وهذه المكونات صُنعت باستخدام تكنولوجيا معينة ولنقل T، فسيُقيد تصميم المعمارية الداخلية بسمات التكنولوجيا T.

وفي الوقت ذاته، يمكن أن يتشكَّل تصميم المعمارية الخارجية ويتقيد بطبيعة المعمارية الداخلية والعكس صحيح. ومن ثَم، توجد علاقة حميمة بين البيئة الحوسبية والمعمارية الخارجية للكمبيوتر والمعمارية الداخلية له والتكنولوجيا المادية (انظر الشكل ٥-١).



شكل ٥-١: معماريات الكمبيوتر وقيودها الخارجية.

المعمارية الخارجية

«قدسُ الأقداس» لمعمارية الكمبيوتر الخارجية هي «مجموعة التعليمات» الخاصة بها التي تحدِّد مجموع العمليات التي يمكن أن يعطي المبرمِج التعليمات للكمبيوتر أن ينفِّذها

مباشرةً. وأنواع العمليات التي يمكن تنفيذها بالتحديد ستُحدَّد وكذلك ستُحدَّد بمجموعة «أنواع البيانات» التي يدعمها أو «يتعرَّف عليها» الكمبيوتر مباشرةً. على سبيل المثال، إذا كان الهدف من الكمبيوتر أن يدعم العمليات الحوسبية العلمية والهندسية دعمًا فعالًا، فسيكون نوعا البيانات المهمان الأعداد الحقيقية (مثل 6.483، و4×10^8، و-0.000021، وغيرها) والأعداد الصحيحة. ومن ثَم ينبغي أن تتضمَّن مجموعة التعليمات نطاقًا من التعليمات الحسابية.

بالإضافة إلى هذه التعليمات المحدَّدة بالمجال، ستكون هناك دائمًا مجموعة تعليمات ذات أغراض عامة، على سبيل المثال لتنفيذ أنواع تراكيب لغة البرمجة الدالة على الشرط (مثل if then else) والدالة على التكرار (مثل while do) والدالة على التفريع غير الشرطي (مثل goto). وتوجد تعليمات أخرى قد تمكِّن البرنامج من أن ينتظم في مقاطع أو وحدات مسئولة عن أنواع مختلفة من العمليات الحوسبية، مع إتاحة القدرة على نقل التحكم من وحدة إلى أخرى.

في الواقع، التعليمة عبارة عن «قالب» يصف العملية المراد تنفيذها بالإضافة إلى المراجع الخاصة بمواقع («عناوين») عناصر بياناتها المُدخلة («المعاملات» حسب الاصطلاح في مجال معمارية الكمبيوتر)، والموقع الذي ستوضع فيه بيانات المخرجات الخاصة بها. وتقتضي فكرة «العنوان» هذه مساحة «ذاكرة». ومن ثَم توجد مكونات ذاكرة تعدُّ جزءًا من المعمارية الخارجية. إضافة إلى ذلك، تشكِّل هذه المكونات تسلسلًا هرميًّا بوجه عام:

الذاكرة الطويلة الأجل: تُعرف باسم «المخزن المساعد» أو «الذاكرة الثانوية» أو «القرص الصلب».

الذاكرة المتوسطة الأجل: وتُعرف أيضًا باسم «الذاكرة الأساسية».

الذاكرة القصيرة الأجل للغاية (العاملة): وتُعرف باسم السجلات.

هذا تسلسل هرمي من حيث الاحتفاظ بالمعلومات وسعة الحجم وسرعة الوصول. ومن ثَم على الرغم من أن المعمارية الخارجية مجرَّدة، فإن الوجه المادي في جهاز الكمبيوتر المادي أصبح واضحًا؛ بمعنى أن المساحة (سعة الحجم) والوقت من الأنماط المادية حيث إنهما يُقاسان بوحدات مادية (وحدات البت أو البايت للمعلومات، والنانوثانية أو البيكوثانية للوقت، وغيرها) وليس بوحدات مجردة. هذا المزيج هو ما يجعل معمارية الكمبيوتر (الخارجية والداخلية) أداة حدية.

الذاكرة الطويلة الأجل هي أطول الذاكرات احتفاظًا بالمعلومات (أي، في قدرتها على «التذكر») حيث إنها ذاكرة دائمة وتصلح لكل الأغراض العملية. وهي الأكبر كذلك من حيث سعة الحجم، ولكنها الأبطأ في سرعة الوصول إليها. أما الذاكرة المتوسطة الأجل فلا تحتفظ بالمعلومات إلا إذا كان الكمبيوتر قيد التشغيل. فهي تفقد المعلومات عند إيقاف تشغيل الكمبيوتر. سعة حجم هذه الذاكرة أقلُّ بكثير من الذاكرة الطويلة الأجل، ولكن وقت الوصول إليها أقصر بكثير من الذاكرة الطويلة الأجل. وقد تغيِّر الذاكرة القصيرة الأجل أو العاملة محتوياتها عدة مرات في سياق العملية الحوسبية الواحدة؛ فسعة حجمها أقل بعدة أمثال من الذاكرة المتوسطة الأجل، ولكن وقت الوصول إليها أقصرُ بكثير جدًّا من الذاكرة الطويلة الأجل.

السبب في وجود التسلسل الهرمي في الذاكرة هو الحفاظ على توازن جيد بين متطلبات الاحتفاظ بالمعلومات والمساحة ووقت الوصول من أجل العمليات الحوسبية. كذلك ستحتوي مجموعة التعليمات على تعليماتٍ لتنفيذ عمليات نقل البرامج والبيانات بين مكونات الذاكرة هذه.

تقوم السمات الأخرى للمعمارية الخارجية على مجموعة التعليمات ومجموعة أنواع البيانات الخاصة بها. على سبيل المثال، يجب أن تتضمن التعليمات طرقًا لتحديد المواقع (العناوين) في ذاكرة المعاملات والتعليمات. ويطلق على الطرق المختلفة لتحديد عناوين الذاكرة «أنماط العنونة». سيكون هناك أيضًا قواعد أو أعراف لتنظيم وتشفير التعليمات الخاصة بمختلف أنواع البيانات بحيث يمكن حفظها داخل الذاكرة بفاعلية؛ ويطلق على هذه الأعراف «تنسيقات التعليمات». وبالمثل، «تنسيقات البيانات» عبارة عن قواعد لتنظيم أنواع البيانات المتنوعة؛ حيث تُحفظ عناصر البيانات من نوع معيَّن من البيانات في الذاكرة طبقًا لتنسيق البيانات ذي الصلة.

وأخيرًا، هناك عنصر مهم في المعمارية الخارجية وهو «طول الكلمة». يحدد هذا العنصر كمَّ المعلومات (الذي يقاس بوحدات البت) الذي يمكن قراءته من الذاكرة المتوسطة الأجل (الأساسية) أو نسخه إليها «في آنٍ واحد». إن سرعة تنفيذ التعليمات تعتمد اعتمادًا كبيرًا على طول الكلمة، وكذلك نطاق البيانات الذي يمكن الوصول إليه لكل وحدة من الزمن.

فيما يلي أمثلة نموذجية على تعليمات الكمبيوتر (أو تعليمات الآلة، وهو مصطلح قد استخدمته من قبل) مكتوبة بأسلوب (لغة تجميع) رمزي، بالإضافة إلى معانيها (أي، الإجراءات التي تتسبب هذه التعليمات في تنفيذها).

التعليمة	المعنى (الإجراء)
(1) LOAD R2, (R1, D)	R2 ← main-memory [R1 + D]
(2) ADD R2, 1	R2 ← R2 + 1
(3) JUMP R1, D	**goto** main-memory [R1 + D]

المفاتيح:
R1 وR2: سجلان
main-memory: ذاكرة متوسطة الأجل
1: ثابت العدد الصحيح «1»
D: عدد صحيح

تضيف R1 + D في التعليمة الأولى العدد الصحيح D إلى «محتويات» لـ R1 وهذا يحدِّد عنوان مُعامل في الذاكرة الأساسية. أما في التعليمة الثالثة R1 + D فتحسب عنوانًا كذلك، ولكن يفسَّر هذا العنوان على أنه خاص بتعليمةٍ ينتقل إليها التحكم في الذاكرة الأساسية.

المعمارية الداخلية

الكمبيوتر المادي في محصِّلته عبارة عن مجموعة من الدوائر الإلكترونية والأسلاك وغيرها من المكونات المادية. ومن حيث المبدأ، يمكن شرح المعمارية الخارجية باعتبارها نتاجًا لهيكل هذه المكونات المادية وسلوكها. لكن المسافة المفاهيمية بين أداة مجردة مثل المعمارية الخارجية والدوائر الإلكترونية المادية كبيرةٌ للغاية لدرجة أن محاولة القيام بهذا الشرح ليست أجدى من محاولة شرح أو وصف كائن حي كامل (ربما باستثناء البكتيريا والفيروسات) من حيث بيولوجيا خلاياه. ما نعنيه أن بيولوجيا الخلية لا تفي مثلًا بشرح بنية جهاز القلب والأوعية الدموية ووظائفه. فلا بد من فهْم الكيانات التي تعلو مستوى الخلايا (مثل الأنسجة والأعضاء) قبل فهْم الجهاز ككل. ومن هذا المنطلق أيضًا، لا تفي نظريةُ الدوائر الإلكترونية الرقمية بشرح المعمارية الخارجية للكمبيوتر، سواء كان كمبيوترًا محمولًا أو حتى أقوى كمبيوتر فائق على مستوى العالم.

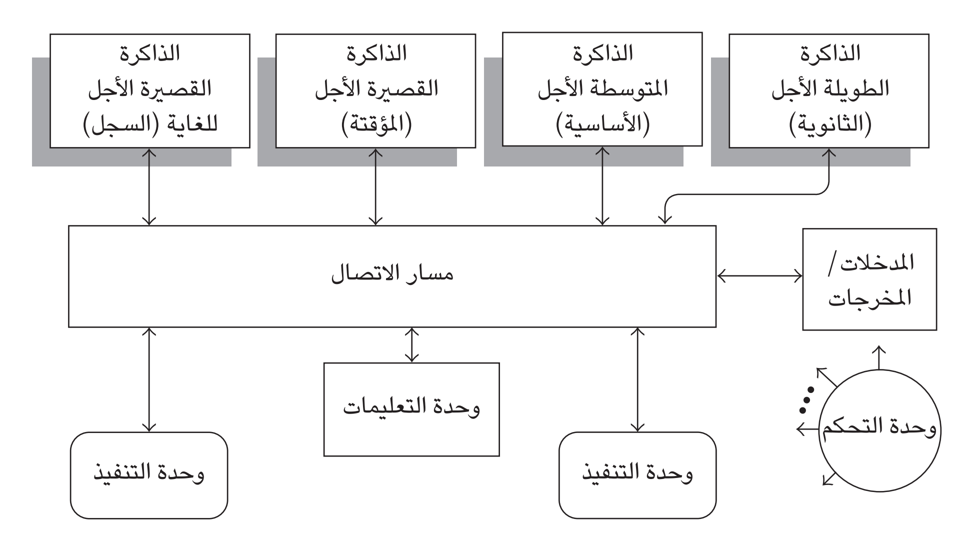
هذه المسافة المفاهيمية في حالة أجهزة الكمبيوتر — والتي يطلق عليها في بعض الأحيان «الفجوة الدلالية» — تُقرَّب بطريقة التسلسل الهرمي. فتنفيذ المعمارية الخارجية يُوضح من خلال المعمارية الداخلية ومكوناتها. وإذا كانت المعمارية الداخلية نفسها معقَّدة ولا تزال توجد مسافة مفاهيمية بينها وبين الدوائر الإلكترونية، فإنها توصف وتُشرح من خلال مستوى تجريد أقل يطلق عليه «المعمارية الدقيقة». يمكن تفصيل المعمارية الدقيقة بدورها إلى ما يسمَّى «مستوى المنطق»، وقد يكون هذا المستوى قريبًا بالدرجة الكافية من مستوى الدوائر الإلكترونية بحيث يمكن تنفيذه من خلال مكونات مستوى المنطق. وبوجه عام، سينطبق على الكمبيوتر المادي مستويات الوصف/التجريد التالية:

المستوى ٤: المعمارية الخارجية
المستوى ٣: المعمارية الداخلية
المستوى ٢: المعمارية الدقيقة
المستوى ١: مستوى المنطق
المستوى ٠: مستوى الدوائر الإلكترونية

وبوجه عام، يهتم اختصاصيو معمارية الكمبيوتر بالمعماريات الخارجية والداخلية، وكذلك تفصيل المعمارية الداخلية الذي ذكرناه من قبل باسم المعمارية الدقيقة (والذي سنوضحه لاحقًا). إنهم لا يهتمون بالسمات التي تشكل هذه المستويات المعمارية فحسب، بل يهتمون بالعلاقة فيما بينها أيضًا.

المكونات الأساسية لمعمارية الكمبيوتر الداخلية موضَّحة في الشكل ٥-٢. وهي تتكوَّن من الأجزاء التالية. أولًا، «نظام الذاكرة» ويتضمن التسلسل الهرمي للذاكرة الظاهر في المعمارية الخارجية والمتضمن أيضًا مكونات أخرى لا تُرى إلا في مستوى المعمارية الداخلية. يحتوي هذا النظام على وحدات تحكُّم مسئولة عن إدارة المعلومات (البنيات الرمزية) التي تمر بين الذاكرات في التسلسل الهرمي وبين النظام وباقي مكونات الكمبيوتر. ثانيًا، وحدة واحدة أو أكثر من «وحدات تفسير التعليمات» التي تجهز التعليمات من أجل تنفيذها والتحكم في تنفيذها. ثالثًا، وحدة واحدة أو أكثر من «وحدات التنفيذ» المسئولة عن التنفيذ الفعلي لفئات التعليمات المتنوعة المطلوبة في عملية الحوسبة. (والاسم الجامع لكلٍّ من نظام (أنظمة) تفسير التعليمات ووحدة (وحدات) التنفيذ هو «معالج» الكمبيوتر.) رابعًا، «شبكة الاتصال» التي تنقل بنيات الرموز فيما

بين المكونات الوظيفية الأخرى. خامسًا، «نظام المدخلات/المخرجات» المسئول عن استقبال بنيات الرموز من — وإرسالها كذلك إلى — بيئة الكمبيوتر المادي. وأخيرًا، «وحدة التحكم» المسئولة عن إصدار الإشارات التي تتحكم في أنشطة المكونات الأخرى.



شكل ٥-٢: مخطط المعمارية الداخلية للكمبيوتر.

يمكن تشبيه وحدات التنفيذ بأعضاء جسد الكائن الحي. يمكن أن تكون هذه الوحدات عالية التخصص ولا تنفِّذ سوى أنواع معينة من العمليات على أنواع محددة من البيانات، ويمكن أن تكون ذات أغراض أشملَ، ومن ثَم تستطيع تنفيذ مجموعات شاملة من العمليات. على سبيل المثال، يمكن تخصيص وحدة تنفيذ بحيث لا تنفِّذ سوى العمليات الحسابية ذات الأعداد الصحيحة، في حين تختص وحدة أخرى بالعمليات الحسابية ذات الأعداد الحقيقية، ويمكن ألا تنفِّذ وحدة ثالثة شيئًا سوى معالجة سلاسل وحدات البت بطرقٍ متعددة، فيما تنفذ وحدة أخرى رابعة العمليات على سلاسل الرموز، وهكذا.

ومن حيث المكونات الداخلية، سيكون للمعالج عناصر ذاكرته الخاصة المتناهية قصر الأجل أو «العابرة» (حيث إنها تحتفظ بالمعلومات لمدة أقصر من السجلات المرئية في المعمارية الخارجية، ويطلق عليها أحيانًا «السجلات الانتقالية») والتي تُحضر إليها

المعلومات من الذاكرات الأخرى قبل أن تعالج من خلال وحدات تفسير التعليمات أو معالجتها. تشكِّل السجلات الانتقالية المستوى «الأدنى» في التسلسل الهرمي للذاكرة المرئي في المعمارية الداخلية.

لا يزال هناك مكوِّن آخر في التسلسل الهرمي للذاكرة المرئي في المعمارية الداخلية، ولكن (عادةً) ما يكون مخفيًّا في المعمارية الخارجية. يسمَّى هذا العنصر بالذاكرة «المؤقتة» ويقع بين الذاكرة المتوسطة الأجل (الأساسية) والذاكرة القصيرة الأجل للغاية (السجل). في الشكل ٥-٢، هذا العنصر موضَّح باسم «الذاكرة القصيرة الأجل». تقع سعة حجم هذه الذاكرة وسرعة الوصول إليها بين سعة الحجم وسرعة الوصول الخاصين بهذين النوعين من الذاكرة. الفكرة الأساسية للذاكرة المؤقتة هي بما أن التعليمات داخل وحدة البرنامج تُنفَّذ على نحوٍ متسلسل (عادةً)، فيمكن وضع جزء من التعليمات في الذاكرة المؤقتة بحيث يمكن الوصول إليه بسرعةٍ أكبر مما لو كان في الذاكرة الأساسية. وبالمثل، طبيعة سلوك البرنامج هي أنه غالبًا ما يتم الوصول إلى البيانات أيضًا من عناوين متسلسلة في الذاكرة الأساسية؛ لذلك يمكن أيضًا وضع أجزاء من البيانات في ذاكرة مؤقتة. ولن يتم الوصول إلى الذاكرة الأساسية إلا في حال عدم وجود التعليمات أو عنصر البيانات ذي الصلة في الذاكرة المؤقتة، وسيؤدي ذلك إلى تبديل الجزء الجديد من المعلومات ذات الصلة بالجزء الموجود في الذاكرة المؤقتة، بحيث تكون المرجعية المستقبلية للتعليمات والبيانات متاحة في الذاكرة المؤقتة.

«كمبيوتر داخل الكمبيوتر»

إذن كيف يمكن أن نصل المعمارية الخارجية بالمعمارية الداخلية؟ كيف ترتبطان بعضهما ببعض في الواقع؟ لفهم هذه العلاقة، يجب أن نفهم وظيفة «وحدة التحكم» (المكتوبة على نحو منعزل مهيب في الشكل ٥-٢ وكأنها صندوق أسود).

وحدة التحكم هي دماغ الكمبيوتر مجازًا، وكأنها كمبيوتر مصغَّر، وهي توصف في بعض الأحيان بأنها «كمبيوتر داخل الكمبيوتر». إنها العضو الذي يدير كل أنشطة الأنظمة الأخرى وحركة بنيات الرموز فيما بينها، ويتحكم فيها ويسلسلها. وهي تقوم بذلك عن طريق إصدار «إشارات تحكُّم» (وهي بنيات رموز مختلفة في فئتها عن التعليمات والبيانات) إلى الأجزاء الأخرى في الجهاز متى وكيفما اقتضت الحاجة. إنها محرِّك الدمى الذي يحرك الخيوط لتنشيط الأنظمة الأخرى التي تشبه الدمى.

وعلى وجه الخصوص، تصدر وحدة التحكم إشاراتٍ إلى المعالج (الذي هو مزيج من وحدات تفسير التعليمات وتنفيذها) لتجعل المعالج ينفِّذ خوارزمية تكرارية يطلق عليها عادةً «دورة التعليمات». دورة التعليمات هي التي تربط المعمارية الخارجية بالمعمارية الداخلية. وفيما يلي الصيغة العامة لهذا:

ICYCLE:

Input: *main-memory:* medium-term memory; *registers:* short-term memory;

Internal: *pc:* transient buffer; *ir:* transient buffer; *or:* transient buffer {pc, short for 'program counter', holds the address of the next instruction to be executed; *ir*, 'instruction register', holds the current instruction to be executed; *or* will hold the values of the operands of an instruction}.

FETCH INSTRUCTION: Using value of pc transfer instruction from main-memory to *ir* (*ir* ← main-memory [*pc*])

DECODE the operation part of instruction in *ir*;

CALCULATE OPERAND ADDRESSES: decode the address modes of operands in instruction in *ir* and determine the effective addresses of operands and result locations in main-memory or registers.

FETCH OPERANDS from memory system into *or*;

EXECUTE the operation specified in the instruction in *ir* using the operand values in *or* as inputs.

STORE result of the operation in the destination location for result specified in *ir*.

UPDATE PC: if the operation performed in EXECUTE is not a **goto** type operation then *pc* ← *pc* + 1. Otherwise do nothing: EXECUTE will have placed the address of the target **goto** instruction in *ir* into *pc*.

وحدة التحكم هي التي تتحكم في دورة التعليمات، ولكن وحدة تفسير التعليمات هي التي تنفِّذ الخطوات من خطوة إحضار التعليمة إلى خطوة إحضار المعاملات في الدورة، ثم خطوتَي التخزين والتحديث، وستنفِّذ وحدة التنفيذ خطوة التنفيذ. لنضرب مثلًا محددًا بتعليمة LOAD المذكورة فيما سبق:

$$\text{LOAD R2, (R1, D)}$$

لاحظ أن دلالة هذه التعليمة على مستوى المعمارية الخارجية ببساطة هي:

$$\text{R2} \leftarrow \text{Main}-\text{memory [R1 + D]}$$

وعلى مستوى «المعمارية الداخلية»، فإن تنفيذ التعليمة ينطوي على أداء دورة التعليمات. تُحضر التعليمة إلى ir، ثم يُفك تشفيرها وتُحسب عناوين المعاملات، ثم تُحضر المعاملات وتُنفَّذ التعليمة، وتُخزَّن النتيجة في السجل R2. وتُجرَّد كلُّ خطواتِ دورةِ التعليمات هذه في المعمارية الخارجية باعتبارها تفاصيلَ غير ضرورية، وذلك بقدرِ ما يخص مستخدِمي المعمارية الخارجية (مبرمجي النظام).

البرمجة الدقيقة

من باب التأكيد، دورة التعليمات عبارة عن خوارزمية تخضع خطواتها لسيطرة وحدة التحكم. في الواقع، قد لا يصعب على المرء فهْم أن هذه الخوارزمية يمكن تنفيذها «باعتبارها برنامجًا تنفِّذه وحدة التحكم» مع باقي الكمبيوتر (نظام الذاكرة ووحدة تفسير التعليمات ووحدات التنفيذ ومسارات الاتصال ونظام المدخلات/المخرجات) باعتباره جزءًا من «بيئة» البرنامج. مبتكِر هذه الرؤية — وكذلك تصميم معمارية وحدة التحكم بناءً عليها — هو عالِم الكمبيوتر البريطاني موريس ويلكس، وقد أطلق عليها اسم «البرمجة الدقيقة». وهي تشير إلى أن وحدة التحكم المبرمَجة بشكلٍ دقيق تنفِّذ برنامجًا دقيقًا ينفِّذ دورة التعليمات لكل نوع مميز من التعليمات، ما جعل البعض يطلق على وحدة التحكم المبرمَجة بشكل دقيق اسم «كمبيوتر داخل الكمبيوتر». وفي الحقيقة، معمارية الكمبيوتر كما يراها «اختصاصي البرمجة الدقيقة» تكون بالضرورة مفصَّلة أكثرَ من المعمارية الداخلية المشار إليها في الشكل ٥-٢. ورؤية اختصاصي البرمجة الدقيقة (أو منفذ وحدة التحكم) هذه للكمبيوتر هي «المعمارية الدقيقة» التي ذكرناها من قبل.

الحوسبة المتوازية

من الخصال المتأصلة في مبدعي الأشياء — مثل المهندسين والفنانين والحرفيين والكتَّاب وغيرهم — عدم الرضا أبدًا عما أنجزوه؛ فهم لا يفتئون يرجون صناعة أدواتٍ أفضل (مهما كان معيار «الأفضلية»). وفي مجال صناعة أجهزة الكمبيوتر المادية، فإن الرغبتين المهيمنتين متعلقتان بالمساحة والوقت؛ بمعنى صناعة أجهزة أصغر وأسرع.

من استراتيجيات تحقيق هذين الهدفين تحسين التكنولوجيا المادية. وتتعلَّق هذه الاستراتيجية بفيزياء الحالة الصلبة والإلكترونيات وتكنولوجيا التصنيع وتصميم الدوائر الإلكترونية. ولا يخفى على أحد ممن يستخدمون أجهزةَ الكمبيوتر المحمولة والأجهزة اللوحية والهواتف الذكية هذا التقدُّم الاستثنائي على مدار ستين سنة أو نحو ذلك منذ صناعة أول دائرة متكاملة، ما وفَّر إمكانية تصنيع مكونات ذات كثافة أعلى وحجم أصغر بالإضافة إلى التركيز المستمر على زيادة قوة الحوسبة. هناك فرضية شهيرة تسمى بقانون مور — نسبة إلى اسم مبتكِرها المهندس الأمريكي جوردون مور — تقول بأن كثافة مكونات الدوائر الأساسية على شريحة واحدة تتضاعف كلَّ عامين تقريبًا، وهو ما أُثبِت تجريبيًّا على مرِّ السنين.

ولكن في ضوء التقدم الحادث في مجال التكنولوجيا المادية، نجد أن اختصاصيي معمارية الكمبيوتر طوَّروا أساليبَ لزيادة «سعة المعالجة» أو «السرعة» الخاصة بالعمليات الحوسبية، والتي تقاس بمقاييس مثل عدد التعليمات التي تتم معالجتها لكل وحدة من الزمن أو عدد بعض العمليات الهامة (مثل العمليات الحسابية باستخدام الأعداد الحقيقية إذا كان الكمبيوتر مخصَّصًا للعمليات الحوسبية العلمية أو الهندسية) لكل وحدة من الزمن. تندرج هذه الاستراتيجيات المعمارية تحت مسمى «المعالجة المتوازية».

الفكرة الأساسية بسيطة للغاية. يقال إن العمليتين أو المهمتين $T1$ و$T2$ يمكن تنفيذهما بالتوازي إذا حدثتا في «تدفُّق مهام تسلسلي» (مثل التعليمات في برنامج تسلسلي)، وكانتا مستقلتين بشكل متبادل. ويتحقق هذا الاستقلال التبادلي إذا كانتا تستوفيان شروطًا معينة. وستعتمد الطبيعة الدقيقة لهذه الشروط ودرجة تعقيدها على عدة عوامل، وبالأخص:

(أ) طبيعة المهام.

(ب) بنية تدفُّق المهام؛ بمعنى هل يحتوي التدفُّق على عبارات تكرارية (مثل أنواع مهام while do) أو عبارات شرطية (مثل if then else) أو عبارات goto.

(ج) طبيعة الوحدات التي تنفِّذ المهام.

لنضرب المَثل بموقفٍ يتشارك فيه معالجان متطابقان نظامَ ذاكرة واحدًا. نريد أن نعرف الظروف التي يمكن أن يبدأ في ظلها تنفيذ المهمتين T1 وT2 بالتوازي اللتين تَظهران في تدفُّق مهام تسلسلي.

لنفترض أن مجموعة عناصر البيانات المدخلة إلى المهمة T1 أُعطيت الاسم INPUT1 وتلك الخاصة بعناصر البيانات المخرجة منها أُعطيت الاسم OUTPUT1. وبالمثل في المهمة T2، يكون لدينا المدخلات INPUT2 والمخرجات OUTPUT2. ولنفترض أن هذه المدخلات والمخرجات عبارة عن مواقع في الذاكرة الأساسية والسجلات أو أيهما. عندئذٍ، يمكن تنفيذ المهمة T1 والمهمة T2 بالتوازي (ويرمز إليها $T1\|T2$) إذا ما استوفيت كل الشروط التالية:

(أ) استقلال المدخلات INPUT1 عن المخرجات OUTPUT2. (بمعنى أنه لا يوجد شيء مشترك بينهما.)
(ب) استقلال المدخلات INPUT2 عن المخرجات OUTPUT1.
(ج) استقلال المخرجات OUTPUT1 عن المخرجات OUTPUT2.

تُعرف هذه الشروط باسم شروط برنشتاين، نسبةً إلى عالِم الكمبيوتر إيه جيه برنشتاين الذي كان أول مَن صاغها. وإذا لم يُستوفَ شرط واحد منها، فستوجد علاقة اعتمادية للبيانات بينهما، ولن يمكن تنفيذ المهام بالتوازي. لنضرب مثالًا بسيطًا بمقطع من تدفُّق برنامج يتكون من عبارات التعيين التالية:

[1] $A \longleftarrow B + C;$
[2] $\mathrm{D} \longleftarrow \mathrm{B} * \mathrm{F/E};$
[3] $\mathrm{X} \longleftarrow \mathrm{A} - \mathrm{D};$
[4] $\mathrm{W} \longleftarrow \mathrm{B} - \mathrm{F}.$

إذن:

INPUT1 = $\{B, C\}$, OUTPUT1 = $\{A\}$
INPUT2 = $\{B, E, F\}$, OUTPUT2 = $\{D\}$
INPUT3 = $\{A, D\}$, OUTPUT3 = $\{X\}$
INPUT4 = $\{B, F\}$, OUTPUT4 = $\{W\}$

بتطبيق شروط برنشتاين، يتضح التالي: أولًا، يمكن تنفيذ العبارتين الأولى والثانية بالتوازي؛ ثانيًا، يمكن تنفيذ العبارتين الثالثة والرابعة بالتوازي؛ لكن، ثالثًا، يوجد اعتمادية بين البيانات بين العبارتين الأولى والثالثة (المتغير A)؛ رابعًا، يوجد قيد خاص باعتمادية البيانات بين العبارتين الثانية والثالثة (المتغير D)، ومن ثَم لا يمكن تنفيذ هذين الزوجين بالتوازي.

إذن، بأخذ شروط التوازي وعدم التوازي هذه في الاعتبار، وبافتراض أنه يوجد ما يكفي من المعالجات التي يمكن أن تنفِّذ العبارات المتوازية في آن واحد، فإن الترتيب الفعلي لتنفيذ العبارات سيكون بالشكل التالي:

Statement [1] ‖ Statement [2];
Statement [3] ‖ Statement [4].

يوضِّح هذا الترتيب التسلسلي/المتوازي بنيةَ «البرنامج المتوازي»؛ حيث تكون المهام عبارات فردية توصف باستخدام إحدى لغات البرمجة. لكن لننظر في الكمبيوتر المادي ذاته. الهدف من البحث في المعالجة المتوازية له شقَّان واسعا النطاق، يتمثل أولهما في ابتكار خوارزميات أو استراتيجيات يمكن أن تكتشف وجهَ التوازي بين المهام، ومن ثَم تجدول المهام المتوازية أو تعيِّنها إلى وحدات مختلفة لتنفيذ المهام في نظام الكمبيوتر. أما الشق الثاني، فيتمثل في تصميم أجهزة كمبيوتر تدعم المعالجة المتوازية.

من منظور اختصاصي معمارية الكمبيوتر، توجد إمكانية التوازي على عدة مستويات من التجريد. وفيما يلي بعضٌ من «مستويات التوازي» هذه:

(١) تنفيذ تدفقات مهام (أو تعليمات) على نحوٍ متزامن على تدفقات بيانات مستقلة على عدة معالجات منفصلة، ولكن مع اتصال تدفقات المهام بعضها مع بعض (على سبيل المثال، بتمرير الرسائل أو نقل البيانات فيما بينها).

(٢) تنفيذ تدفقات مهام (أو تعليمات) على نحوٍ متزامن على تدفق بيانات فردي مشترك على عدة معالجات داخل كمبيوتر واحد.

(٣) شغل تدفقات بيانات متعددة لوحدات ذاكرة متعددة يتم الوصول إليها بنحو متزامن من خلال تدفق مهام (تعليمات) واحد على معالج واحد.

(٤) تنفيذ مقاطع (وتسمَّى سلاسل) تدفق مهام واحد بشكل متزامن، إما على معالج واحد أو على معالجات متعددة.

(٥) تنفيذ مراحل أو خطوات تعليمة واحدة بنحو متزامن داخل دورة التعليمات.

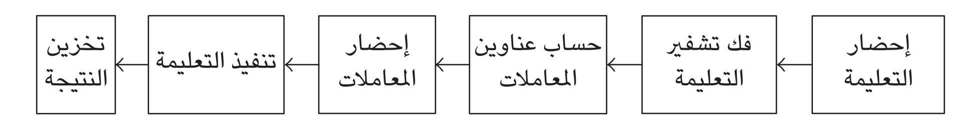
(٦) تنفيذ أجزاء برنامج دقيق بنحو متزامن داخل وحدة تحكم خاصة بجهاز الكمبيوتر.

تستخدم كل معماريات المعالجة المتوازية صورًا متباينة من الاحتمالات المذكورة آنفًا، وكثيرًا ما يجري الجمع بينها.

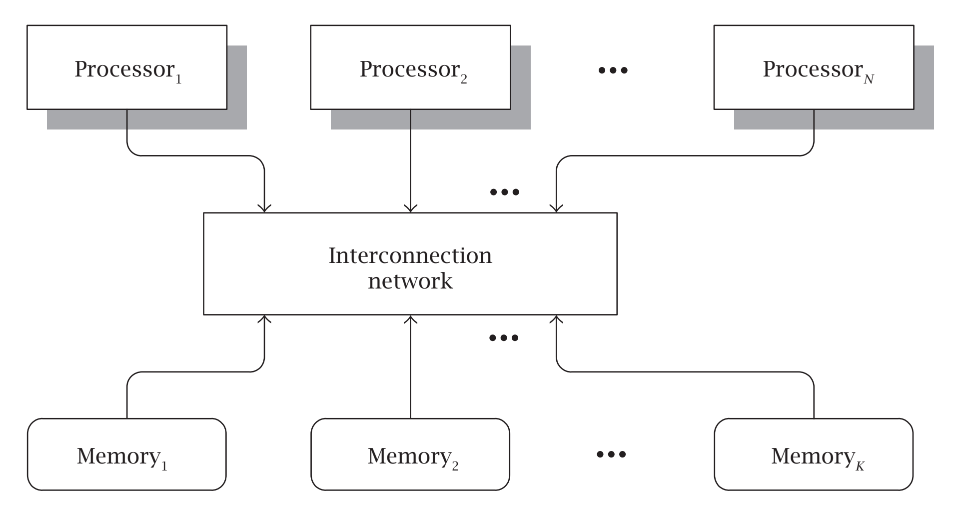
لنضرب مثالًا بمستوى التجريد الخامس المذكور فيما سبق. فكرة هذا المستوى هي أنه بما أن دورة التعليمات تتكون من عدة مراحل (بدايةً من إحضار التعليمة إلى تحديث pc)، فيمكن تنظيم المعالج نفسه الذي ينفِّذ دورة التعليمات في شكل «قناة تجزئة» (pipeline) تتكوَّن من هذه المراحل المتعددة. وستمر التعليمة الواحدة بكل مراحل القناة بطريقة تسلسلية. وتكون «المهام» في هذا المستوى من التجريد هي خطوات دورة التعليمات التي تمر عبْرها التعليمة. ولكن عندما تشغل التعليمة مرحلة من المراحل، تكون المراحل الأخرى خالية ويمكنها أن تعالج المراحل ذات الصلة للتعليمات الأخرى في تدفق التعليمات. في الحالة المثلى، يمكن تنفيذ دورة تعليمات تتكون من سبع خطوات بقناة تجزئة معالج تعليمات مكونة من سبع مراحل، ومن ثَم تصبح كل مراحل القناة مشغولة؛ حيث تعمل على سبع تعليمات مختلفة بالتوازي وكأنها خط تجميع. بالطبع هذه الحالة المثلى. لكن في الواقع العملي، يمكن لأزواج التعليمات في تدفق التعليمات أن تخرق شروط برنشتاين، ومن ثَم يمكن أن تحتوي القناة على مراحل «فارغة»، ويرجع سببُ ذلك إلى قيود اعتمادية البيانات بين مراحل أزواج التعليمات.

ويطلق على المعماريات التي تدعم هذا النوع من المعالجة المتوازية اسم المعماريات ذات قنوات التجزئة (انظر الشكل ٥-٣).

لنضرب مثالًا آخر، انظر إلى المهام في مستوى التجريد الأول المذكور فيما سبق. في هذا المستوى، تنفِّذ عدة معالِجات — داخل الكمبيوتر نفسه — تدفقات التعليمات (التي تنتمي، لنقُل، إلى وحدات برنامج منفصلة) بالتوازي. ربما تصل هذه المعالجات إلى نظام ذاكرة فردي مشترك واحد أو ربما ينقسم نظام الذاكرة ذاته إلى وحدات ذاكرة مستقلة. على أية حال، ستكون «شبكة ربط الذاكرة والمعالج» المتطورة بمثابة الواجهة بين أنظمة الذاكرة والمعالجات (انظر الشكل ٥-٤). ويُطلق على هذه المخططات «المعماريات المتعددة المعالِجات».



شكل ٥-٣: قناة تجزئة تعليمة.



شكل ٥-٤: مخطَّط معالج متعدد.

كما ذكرنا من قبل، الهدف من معماريات المعالجة المتوازية هو زيادة «سعة معالجة» أو «سرعة» نظام الكمبيوتر من خلال وسائل معمارية بحتة. ولكن كما هو موضح في المثال الصغير لعبارات التعيين الأربع، هناك حدود لتوازي تدفق المهام بسبب قيود اعتمادية البيانات؛ ومن ثَم، هناك حدود للسرعة التي يمكن تحقيقها في بيئة معالجة متوازية. صيغ هذا الحد صياغةً كمية في ستينيات القرن العشرين على يد مصمِّم أجهزة الكمبيوتر جين أمدال؛ إذ قال إن السرعة المحتملة للكمبيوتر المتوازي المعالجة مقيَّدة بقدرِ الحوسبة الذي لا يمكن تنفيذه بنحو متوازٍ. ومن ثَم يتوقف تأثير زيادة السرعة الناتج عن زيادة عدد وحدات التنفيذ المتوازية بعد حدٍّ معيَّن. ويطلق على هذا المبدأ «قانون أمدال».

العلم في معمارية الكمبيوتر

بعد أن وصل القارئ إلى هذا القسم من الفصل، ربما يسأل: إذا ما سلَّمنا بأن معماريات الكمبيوتر أدواتٌ حدية، فبأي نحوٍ يُعَد هذا المجال من العلوم الاصطناعية؟

للإجابة عن هذا السؤال، لا بد من إدراكِ أن أكثرَ ما يَلفت الانتباه في هذا المجال هو أن مساحة المعرفة فيه تتكون (بالأساس) من مجموعة مبادئ «استدلالية»، وأن التفكير المتَّبع في تصميم معماريات الكمبيوتر هو «التفكير الاستدلالي».

الاستدلال — الذي مقابله الإنجليزي heuristics مشتَق من الكلمة اليونانية hurisko التي تعني «أن يجد» — هو القواعد أو الافتراضات التي تقدِّم أملًا أو بشرى بحلولٍ لأنواع معيَّنة من المسائل (التي سنناقشها في الفصل التالي)، ولكنها «غير مضمونة النجاح». ولنُعِد صياغةَ ما قاله عالِم الرياضيات الأمريكي من أصلٍ مجريٍّ جورج بوليا الذي أكَّد على نحو شهير دورَ الاستدلال في الاكتشافات الرياضية؛ التفكير الاستدلالي ليس قاطعًا ولا باتًّا ولا مؤكدًا البتة؛ بل هو بالأحرى مؤقَّت ومقبول منطقيًّا وتجريبي.

وكثيرًا ما نُضطر لاستخدام الاستدلال لأنه قد لا يوجد أيُّ خيار آخر. فنركن إلى التفكير الاستدلالي عند غياب المبادئ القائمة على النظريات والمؤكدة والصورية أكثر. ومبدأ فرِّق تسُد الذي تناولناه في الفصل الثالث مثالٌ على أحد المبادئ الاستدلالية المستخدَمة بكثرة في حل المسائل واتخاذ القرارات. إنه مبدأ مقبول يُتوقع منه أن يساعد في حل مسألة معقَّدة ولكن لا يُضمن نجاحه في حالة بعينها. كذلك المعرفة التجريبية مصدر للعديد من أشكال الاستدلال. وتُعدُّ قاعدة «إذا كانت السماء ملبَّدة بالغيوم، فخُذ معك مظلة» مثالًا في هذا الإطار. قد يكون هناك ما يبرِّر أخذ المظلة، ولكن ليس دائمًا.

واستخدام الاستدلال يستجلب معه ضرورةَ «التجريب». فبما أن الاستدلال ليس مضمون النجاح، يكون الملجأ الوحيد تطبيقَه على مسألةٍ بعينها ومعرفةَ ما إذا كان مجديًا عمليًّا؛ أي، القيام بتجربة. وفي المقابل، قد تُشتق مبادئ الاستدلال ذاتها بناءً على تجارب سابقة. «إن الاستدلال والتجارب يسيران جنبًا إلى جنب» هي رؤية استوعبها جيدًا روادُ التفكير الاستدلالي أمثال ألين نيويل وهيربرت سايمون، وكذلك رواد مجال تصميم الكمبيوتر أمثال موريس ويلكس.

وكل هذا مقدِّمة لما يلي: «يندرج مجال معمارية الكمبيوتر ضمن العلوم الاصطناعية التجريبية الاستدلالية».

على مدار العقود التي تلت ابتكارَ الكمبيوتر الرقمي الإلكتروني، ظهرت مجموعةٌ من القواعد والمبادئ والضوابط والافتراضات والمخططات بشأن تصميم معماريات الكمبيوتر،

وتكاد تكون جميعها استدلالية في طبيعتها. ومن الأمثلة على ذلك فكرةُ النظر إلى التسلسل الهرمي للذاكرة باعتباره أحدَ مبادئ التصميم. ومبدأ قنوات التجزئة مثالٌ آخر. إن هذه المبادئ تنشأ من المعرفة التجريبية وعَقد المقارنات والملاحظات المنطقية السليمة.

على سبيل المثال، أدَّت التجارب مع تصاميم المعماريات السابقة والصعوبات التي واجهت إنتاجَ لغة الآلة باستخدام البرامج المترجِمة إلى صياغة مبادئ استدلالية للتغلب على هذه الصعوبات. وفي ثمانينيات القرن العشرين، طرح عالِم الكمبيوتر ويليام وولف عددًا من هذه المبادئ بناءً على خبرة في تصميم البرامج المترجِمة لأجهزة كمبيوتر معينة. وفيما يلي بعضها:

الانتظام. إذا تحقَّقت سِمة (معمارية) معينة بطريقةٍ معينة في أحد أجزاء المعمارية، فمن المفترض تحقيقها بالطريقة ذاتها في كل الأجزاء.

فصل الاهتمامات. (فرِّق تسُد.) ينبغي أن تكون المعمارية ككل قابلةً للتقسيم إلى عدد من السمات المستقلة، بحيث يمكن تصميم كل ميزة على حدة.

القابلية للتركيب. بحكم المبدأين السابقين، ينبغي أن يكون ممكنًا تركيب السمات المستقلة المنفصلة بطرق عشوائية.

لكن لا بد أن تدمج التجارب مبادئ الاستدلال في التصميم. قد تنطوي هذه التجارب على تنفيذ «نموذج أولي» أو آلة تجريبية وإجراء الاختبارات عليها. أو قد تتضمن إنشاءَ نموذج محاكاة (برمجي) للمعمارية وإجراء التجارب على المعمارية المحاكية.

وفي كلتا الحالتين، قد تكشف التجارب عن عيوب في التصميم، وعندئذٍ ستكون النتيجة تعديل التصميم عن طريقِ نبذِ بعض المبادئ وإدخال مبادئ أخرى، ثم تكرار دورة التجريب والتقييم والتعديل.

بالطبع يكاد هذا المخطط يطابق نموذجَ حل المسائل العلمية الذي طرحه فيلسوف العلم كارل بوبر:

$$P1 \longrightarrow TT \longrightarrow EE \longrightarrow P2$$

هنا، تعبِّر $P1$ عن موقف المسألة الحالي، وتعبِّر TT عن النظرية المؤقتة المطروحة لشرح الموقف أو حله، وتعبِّر EE عن عملية التخلص من الأخطاء المتعلقة بالنظرية TT (عن طريق التجارب و/أو التفكير النقدي)، وتعبِّر $P2$ عن موقف المسألة الجديد الناتج

بعد التخلص من الأخطاء. في سياق معمارية الكمبيوتر، تعبِّر $P1$ عن مسألة التصميم المحدَّدة من حيث الأهداف والمتطلبات التي يجب أن يستوفيَها الكمبيوتر النهائي، وتعبِّر TT عن التصميم ذاته القائم على الاستدلال (وهو نظرية الكمبيوتر)، وتعبِّر EE عن عملية إجراء التجارب على التصميم وتقييمه وكذلك التخلُّص من عيوب التصميم ومشكلاته، ويعبِّر الناتج $P2$ عن مجموعةٍ ربما تكون معدَّلة من الأهداف والمتطلبات، والتي تشكِّل مسألة تصميم جديدة.

الفصل السادس

الحوسبة الاستدلالية

ليس للعديد من المسائل حلول خوارزمية. فعند تعليم أحد الأبوين الطفلَ ركوبَ الدراجة، فإنه لا يستطيع أن يقدِّم له خوارزمية يمكنه أن يتعلمها ويطبِّقها مثلما يفعل حينما يتعلم طريقةَ ضرب عددين. كذلك لا يستطيع معلِّم الكتابة الإبداعية أو الرسم أن يقدِّم للطلاب خوارزميات لكتابة قصة واقعية جذابة أو رسم لوحة قماشية تعبيرية تجريدية.

يعود السبب في ذلك جزئيًّا إلى جهل المرء (وحتى لو كان أستاذًا متخصصًا في الكتابة الإبداعية) أو عدم فهمه للطبيعة الدقيقة لتلك المهام. يريد الرسام أن يصوِّر على سبيل المثال ملمسَ رداءٍ مُخملي، وصلابة تفاحة، وسر ابتسامة. لكن الشيء الذي يشكِّل ذلك الملمس المُخملي وتلك الصلابة وتلك الابتسامة الغامضة من منظور الرسامين ربما يكون مجهولًا أو غير معروف بالتحديد، ومن ثَم لا يمكن ابتكار خوارزمية لتصويرها في لوحة. بل إن البعض قد يقول إن الإبداع الفني أو الأدبي أو الموسيقي لا يمكن أبدًا شرحُه باستخدام الخوارزميات.

ثانيًا، تحدِّد الخوارزمية كلَّ الخطوات التي يجب اتباعها. إذ لا يمكننا بناءُ خوارزمية إلا إذا كنا مدركين على نحوٍ واعٍ لخطواتها التي تمثِّل قِوامها. لكن العديد من الإجراءات التي ننفِّذها عند ركوب الدراجة أو التقاط الفروق الدقيقة في المشهد الذي نريد رسمه تحدُث فيما يسميه علماء العلوم المعرفية «اللاوعي المعرفي». ومن ثَم، هناك حدود للمدى الذي يمكن أن ترتقي فيه هذه الأفعال اللاواعية إلى سطح الوعي.

ثالثًا، حتى إنْ فهمْنا طبيعة المهمة فهمًا جيدًا (إلى حدٍّ ما)، فقد تتضمن المهمة متغيرات أو معاملات متعددة تتفاعل بعضها مع بعض بطرق غير بسيطة. وقد تكون معرفتنا بهذه التفاعلات أو فهمنا لها مشوَّشة أو منقوصة أو غير ملائمة بشدة. على سبيل المثال، مسألة تصميم معمارية الكمبيوتر الخارجية (ارجع إلى الفصل الخامس)

تُوضح هذه النقطة. قد يفهم اختصاصي معمارية الكمبيوتر الأجزاء الفعلية التي تكوِّن المعمارية الخارجية (أنواع البيانات، والعمليات، ونظام الذاكرة، وأنماط عنونة المعاملات، وتنسيقات التعليمات، وطول الكلمة) فهمًا جيدًا، لكن نطاق التباينات لكل جزء من هذه الأجزاء وتأثيرات هذه التباينات بعضها على بعض قد يكون مفهومًا فهمًا غيرَ دقيق أو غير واضح. في الواقع، إن فهْم الطبيعة الكاملة لهذه التفاعلات قد يتجاوز القدرات المعرفية لدى هذا الاختصاصي.

رابعًا، حتى إذا كان المرء يفهم المسألة جيدًا بالقدْر الكافي ويمتلك المعرفة بشأن مجال المسألة ويمكنه ابتكار خوارزمية لحلها، فقد لا يمكن توفير مقدار الموارد الحوسبية (الوقت أو المساحة) اللازم لتنفيذ الخوارزمية. ومن الأمثلة على ذلك الخوارزميات ذات تعقيد الوقت الأسي (ارجع إلى الفصل الثالث).

لِعِبُ الشطرنج مثال على ذلك. طبيعة المسألة مفهومة فهمًا جيدًا. فاللعبة لها قواعد محدَّدة للحركات المسموح بها، كما أنها لعبة «تامة المعلومات» من حيث إن كل لاعب يستطيع أن يرى كل قِطع الشطرنج على الرقعة على الدوام. كذلك النتائج المحتملة معروفة بدقة: إما أن يفوز اللاعب الأبيض أو يفوز اللاعب الأسود أو يتعادلان.

لكن لننظر في مأزق اللاعب. كلما أتى دوره للعب، كان هدفه الأسمى اختيار حركة تؤدِّي به إلى الفوز. ومن حيث المبدأ، ثمَّة استراتيجية (خوارزمية) مثلى يمكن أن يتَّبعها اللاعب:

> يفكر اللاعب الذي حلَّ دوره في اللعب في كل الحركات الممكنة لصالحه. ومع كل حركة يفكِّر فيها من تلك الحركات يفكِّر أيضًا في كل الحركات التي يُحتمل أن يقوم بها الخَصم؛ ومع كل حركة محتملة من خَصمه، يفكِّر مرة أخرى في كل حركاته هو المحتملة؛ وهكذا حتى بلوغ الحالة النهائية وهي: إما الفوز أو الخسارة أو التعادل. وبالعمل بطريقة عكسية، يقرر اللاعب إن كان الموقف الحالي سيعزز من الفوز أم لا، ويختار الحركة بناءً على ذلك، وذلك بافتراض أن الخصم يأتي بالحركات الأفضل لصالحه.

تسمَّى هذه الطريقة بطريقة «البحث المستفيض» أو «القوة الغاشمة». وهي تنجح من حيث المبدأ. لكنها «غير عملية» بالطبع. فقد قُدر أنه يوجد حوالي ثلاثين حركة مسموح بها محتملة في النسق النموذجي لرقعة الشطرنج. لنفترض أن مباراة نمطية تستمر حوالي

أربعين حركة قبل أن يستسلم أحد اللاعبين. آنذاك ومنذ بداية المباراة، يجب على اللاعب أن يفكِّر في ثلاثين حركة محتملة تالية، ولكل حركة من هذه الحركات، يوجد ثلاثون حركة محتملة من جانب الخصم؛ وهو ما يساوي 30^2 احتمالًا في الحركة الثانية، ولكل خيار من الخيارات 30^2 هذه، يوجد ثلاثون بديلًا آخر في الحركة الثالثة، وبذلك يصبح لدينا 30^3 احتمالًا. وهكذا حتى يصبح عدد الاحتمالات في الحركة الأربعين 30^{40} احتمالًا. ومن ثَم سيكون على اللاعب في بداية المباراة أن يفكر في $30 + 30^2 + 30^3 + 30^4 + \cdots + 30^{40}$ حركة بديلة قبل أن يختار الحركة «المثلى». هنا يكون مدى المسارات البديلة كبيرًا بدرجة فلكية.

في نهاية المطاف، إذا وجب صياغة خوارزمية لحل مسألةٍ ما، فإن أي معرفة ضرورية بشأن المسألة كي تنجح الخوارزمية «لا بد أن تُدرج بالكامل في الخوارزمية». وكما أشرنا في الفصل الثالث، فإن الخوارزمية جزء قائم بذاته من المعرفة الإجرائية. فلكي نُجريَ اختبار ورقة دَوَّار الشمس، أو نحلَّ مسألة ضرب بالورقة والقلم، أو نقدِّر مضروبَ عددٍ ما، أو ننتج تعبيرات بولندية عكسية من تعبيرات حسابية مرتبة وسطيًّا (ارجع إلى الفصل الرابع)، فكلُّ ما يحتاج المرء إلى معرفته هو الخوارزمية ذاتها. وإن تعذَّر إدراج كل المعلومات الضرورية في الخوارزمية، فلن تكون ثمة خوارزمية.

إن العالم مليء بالمهام أو المسائل التي تتجلى فيها أنواع السمات المذكورة آنفًا. وهي لا تتضمن الأعمال الفكرية أو الإبداعية وحسب — مثل البحث العلمي، والاختراعات، والتصميم، والكتابة الإبداعية، والأعمال الرياضية، والتحليل الأدبي، والبحوث التاريخية — بل تتضمَّن كذلك أنواع المهام التي يقوم بها الممارسون المهنيون — مثل الأطباء، والمهندسين، والمهندسين المعماريين، والمصمِّمين الصناعيين، والمخطِّطين، والمدرسين، والحرفيين. حتى الأنشطة العادية الرتيبة — مثل القيادة وسط زحام مروري، واتخاذ قرار بشأن عرض وظيفي، والتخطيط لرحلة في عطلة — لا تؤدي إلى حلول خوارزمية، أو على الأقل حلول خوارزمية فعالة.

وعلى الرغم من ذلك، يشرع الناس في تنفيذ هذه المهام وحل هذه المسائل. فهم لا ينتظرون الخوارزميات، سواء كانت فعالة أو غير ذلك. في الواقع، لو كان علينا انتظار الخوارزميات كي نحل كل مسائل حياتنا، لانقرض نوعنا منذ زمن بعيد. ومن منظور تطوري، فإن سبل تفكيرنا لا تنطوي على الخوارزميات وحسب. وهذا يطرح السؤال التالي: ما الوسائل «الحوسبية» الأخرى المتاحة لنا كي ننفِّذ هذه المهام؟ الإجابة هي اللجوء إلى نمط حوسبة يستخدم «الاستدلال».

الاستدلال عبارة عن قواعدَ وضوابط ومبادئ وافتراضات قائمة على المنطق السليم والتجربة والحكم على الأشياء والمقارنات والتخمينات المبنية على علم وغير ذلك، والتي تبشِّر بحل المسائل من دون ضمان نجاح حلها. وقد أتينا على ذكر الاستدلال في الفصل السابق عندما تناولنا معمارية الكمبيوتر. لكن الحديث عن معمارية الكمبيوتر باعتبارها علمًا اصطناعيًّا قائمًا على الاستدلال شيء، واستخدام الاستدلال في الحوسبة التلقائية شيء آخر. وهذا الأمر الأخير، «الحوسبة الاستدلالية»، هو ما سنتناوله الآن.

ابحث «وقد» تجد

الحوسبة الاستدلالية تجسِّد روح المغامرة! فهي لا تخلو من عدم اليقين والغموض. فمن يبحث عن حل استدلالي (سواء كان إنسانًا أو كمبيوتر) ليحل مسألةً ما هو فعليًّا كمن يستكشف «أرضًا مجهولة». ومثلما ينخرط الإنسان المتواجد في أرض مجهولة في استكشاف هذه الأرض والبحث فيها، يبحث مَن يقوم بالبحث الاستدلالي عن حل للمسألة — فيما يسميه علماء الكمبيوتر «فضاء المسألة» — وهو ليس على يقين من أنه سيصل إلى الحل. وبذلك يُطلق على أحد أنواع الحوسبة الاستدلالية اسم «البحث الاستدلالي».

لننظر، على سبيل المثال، إلى السيناريو التالي. أنت داخل إلى ساحة كبيرة جدًّا لركن السيارات ملحقة بصالة عرض كبيرة تريد أن تشاهد حدثًا فيها. والمسألة هي العثور على مكان لركن سيارتك. السيارات مصفوفة في أماكنها بالفعل، ولكن يبدو أنك ليس لديك معرفة بتوزيع الأماكن الفارغة أو موقعها. ما الذي يفعله المرء في هذه الحالة؟

في هذه الحالة، تصبح ساحة الركن هي فضاء المسألة حرفيًّا. وكلُّ ما يمكنك فِعله حرفيًّا هو البحث عن مكان فارغ. لكن بدلًا من البحث من دون هدف أو عشوائيًّا، قد تقرر اتباع سياسة «الملاءمة الأولى»؛ بحيث تركن سيارتك في أول مكان تراه فارغًا ومتاحًا. أو يمكنك اتباع سياسة «الملاءمة المثلى»؛ وهي البحث عن مكان فارغ ويكون هو الأقرب إلى صالة العرض.

هاتان السياستان استدلاليتان تساعدان في «توجيه» البحث عبْر فضاء المسألة. بالطبع لا يخلو الأمر من التنازلات: فاستراتيجية الملاءمة الأولى قد توفِّر وقتَ البحث، ولكن قد تضطر إلى أن تسير مسافةً طويلة حتى تصل إلى صالة العرض، أما استراتيجية الملاءمة المثلى فقد تحتاج إلى وقتٍ أطول للبحث لكنها ستختصر وقت السير نسبيًّا إن نجحت. لكنْ بالطبع «السياستان الاستدلاليتان لا تضمنان النجاح»: كلتاهما ليستا خوارزمية بهذا

المعنى. وفي كلتا الحالتين، قد لا يوجد مكان فارغ، وفي هذه الحالة إما أن تظلَّ تبحث وتبحث أو تنهيَ البحث باتباع معيار مختلف مثل «الخروج إن تجاوز وقت البحث الحد المعقول».

لكن العديد من الاستراتيجيات التي تستخدم الاستدلال تتضمن كلَّ سمات الخوارزمية (كما ذكرنا في الفصل الثالث) — لكن هناك فرقٌ واحد ملحوظ وهو أنها لا تعطي سوى إجابات «تكاد تكون صحيحة» للمسألة، أو أنها لا تعطي إجابات صحيحة إلا لبعض صور المسألة. ومن ثَم يشير علماء الكمبيوتر إلى بعض الأساليب الاستدلالية في حل المسائل باسم الخوارزميات «الاستدلالية» أو «التقريبية»، وقد نحتاج في هذه الحالة إلى أن نفرِّق بينها وبين الخوارزميات «الدقيقة». ويشمل المصطلح «الحوسبة الاستدلالية» كلًّا من البحث الاستدلالي والخوارزميات الاستدلالية. وسنعرض مثالًا على الخوارزميات الاستدلالية بعد قليل.

هناك أسلوب استدلالي ذو مستوًى أعلى يسمَّى «القبول بما هو مُرضٍ»

في أي مسألة تحسين، عادةً ما يكون الهدف هو البحث عن أفضل حل ممكن للمسألة. والعديد من مسائل التحسين لها حلول خوارزمية دقيقة. ولسوء الحظ كثيرًا ما تكون هذه الخوارزميات ذات تعقيد وقت أسي، ومن ثَم فمن غير العملي — وحتى من المتعذر — استخدامها في صور كبيرة من المسألة. ومن الأمثلة على ذلك مسألةُ لعبة الشطرنج التي تناولناها من قبل. إذن، ما الذي يجب أن يفعله المكلَّف بحل المسألة إذا كانت الخوارزميات المثلى يتعذر تطبيقها حوسبيًّا؟

بدلًا من السعي العنيد لتحقيق هدف الأمثلية، قد يطمح المكلَّف بحل المسألة إلى تحقيق أهداف ممكنة أو مقبولة أكثرَ تكون أقلَّ من حيث المثالية، ولكنها «جيدة بشكل مقبول». وعند الحصول على حلٍّ يلبي هذا المستوى من الطموح، فسيشعر المكلَّف بحل المسألة بالرضا. وقد صاغ هيربرت سايمون مصطلحًا لهذا النوع من التفكير، ألا وهو «القبول بما هو مُرضٍ» (satisficing). الطموح للوصول إلى حلٍّ مُرضٍ أهونُ من الطموح إلى المثالية؛ إنه اختيار الجيد الممكن على الأمثل غير الممكن.

هذا المبدأ أسلوبٌ استدلالي عام وعالي المستوى للغاية، ويمكن أن يكون بمثابة نقطة انطلاق لتحديد المزيد من الأساليب الاستدلالية الخاصة بالمجال. وبذلك يمكننا أن نطلق عليه «الأسلوب الاستدلالي العالي المستوى».

دعنا نتأمَّل «الأساليب الاستدلالية المُرضية التي لها علاقة بمسألة الشطرنج». لنفكرْ في مأزق لاعب الشطرنج. وكما رأينا، فقد استبعدنا حلَّ البحث الأمثل. ويلزم توافر استراتيجيات عملية أكثر، وهذا يتطلب استخدام مبادئ استدلالية متعلقة بالشطرنج (أي، مرتبطة بالمجال). والاستراتيجية التالية من أبسطها.

لنتأملْ نسقَ رُقعة الشطرنج (مواقع كل قِطع الشطرنج على الرُّقعة حاليًّا) C. ثم علينا تقييم ما إذا كان النسق C واعدًا باستخدام أحد مقاييس «الجودة» G(C) والذي يراعي السمات العامة للنسق C (عدد قِطع الشطرنج وأنواعها، ومواقعها النسبية، وغير ذلك).

لنفترض أن M1، وM2، …، وMn هي الحركات التي يمكن أن يقوم بها اللاعب في النسق C، ولنفترض أن النُّسق الناتجة بعد كل حركة هي M1C، وM2C وهكذا. بعدها نختار الحركةَ التي تزيد من قيمة جودة النسق الناتج. نقصد اختيار الحركة Mi التي تكون قيمةُ جودتها G(MiC) هي الأعلى.

لاحظ أننا حاولنا تحقيقَ نوع من الأمثلية هنا. لكنها محاولة تحسين «موضعية» أو «قصيرة الأجل»؛ حيث إنها تتعلق بحركة تالية واحدة فقط. وليس هذا باستدلال بالغ التعقيد، ولكنه نوع يمكن أن يتحلَّى به اللاعب العادي. لكنه يتطلب بالفعل مستوًى من المعرفة العميقة لدى اللاعب (سواء أكان إنسانًا أم كمبيوترًا) بشأن الجودة النسبية لأنساق الرقعة.

إنَّ لعب الشطرنج يجسِّد صورًا من البحث الاستدلالي المُرضي. أما الآن فلننتقل إلى مثال على الخوارزميات الاستدلالية المُرضية.

الخوارزمية الاستدلالية

تذكَّر الحديث عن المعالجة المتوازية في الفصل السابق. يقول هذا الحديث إن بالإمكان معالجة مهمتين Ti، وTj في تدفُّق مهام (بأي مستوًى من التجريد) بالتوازي بشرط تلبية شروط برنشتاين بشأن استقلالية البيانات.

لندرس الآن تدفقًا تسلسليًّا لتعليمات آلة يولدها برنامج مترجِم لكمبيوتر مادي مستهدف من برنامج تسلسلي مكتوب بلغة عالية المستوى (ارجع إلى الفصل الرابع). لكن إذا كان الكمبيوتر المستهدف ينفِّذ التعليمات بصورة متزامنة، فسيكون للبرنامج المترجِم مهمةٌ أخرى إضافية وهي: تحديد التوازي بين التعليمات في تدفُّق التعليمات وإنتاج

«تدفُّق تعليمات متوازٍ»؛ حيث يتكوَّن كل عنصر من هذا التدفق من مجموعة تعليمات يمكن تنفيذها بالتوازي (لنطلق على ذلك «مجموعة متوازية»).

في واقع الأمر، هذه مسألة تحسين إذا كان الهدف هو تقليل عدد المجموعات المتوازية في تدفُّق التعليمات المتوازي. وكما هو الحال مع مسألة الشطرنج، ستنطوي خوارزمية التحسين على استراتيجية بحث مستفيض، ومن ثَم لن تكون عملية حوسبيًّا.

عمليًّا، ستطبَّق أساليبُ استدلالية مُرضية أكثر. لنضرب مثالًا بخوارزمية أُطلق عليها هنا خوارزمية «الخدمة بأسبقية الوصول».

تأمَّل تدفُّق التعليمات التسلسلي التالي *S*. (ولغرض التبسيط، لا توجد عبارات تكرارية ولا عبارات **goto** في هذا المثال.)

I1: $A \longleftarrow B$;

I2: $C \longleftarrow D + E$;

I3: $B \longleftarrow E + F - 1/W$;

I4: $Z \longleftarrow C + Q$;

I5: $D \longleftarrow A/X$;

I6: $R \longleftarrow B - Q$;

I7: $S \longleftarrow D * Z$.

ستكون خوارزمية الخدمة بأسبقية الوصول كما يلي:

FCFS:

Input: A straight line sequential instruction stream *S*: < I1, I2, . . ., In>;

Output: A straight line parallel instruction stream *P* consisting of a sequence of parallel sets of instructions each and every one of which are present in *S*.

لكل تعليمة متتالية I في التدفق *S* تبدأ بـ I1 وتنتهي بـ In، ضع I في أقرب مجموعة متوازية موجودة ممكنة وتستوفي شروط برنشتاين بشأن استقلال البيانات. وإذا لم يكن ذلك ممكنًا — بسبب اعتمادية البيانات التي تحول دون وضع I في أيٍّ من المجموعات

المتوازية الحالية — يتم إنشاء مجموعة متوازية جديدة (فارغة) «بعد» المجموعات الحالية ووضع I فيها.

وعند تطبيق خوارزمية الخدمة بأسبقية الوصول هذه على المثال السابق، نرى أن المخرجات هي تدفُّق التعليمات المتوازي P:

I1 ‖ I2;

I3 ‖ I4 ‖ I5;

16 ‖ 17.

هنا، يعبِّر الرمز ‖ عن المعالجة المتوازية، ويعبِّر الرمز ; عن المعالجة التسلسلية. وبذلك، يصبح لدينا تدفُّق متوازٍ من ثلاث مجموعات متوازية من التعليمات.

إن خوارزمية الخدمة بأسبقية الوصول استراتيجية مُرضية. فهي تضع كلَّ تعليمة في أقرب مجموعة متوازية ممكنة بحيث تظهر أيضًا التعليمات اللاحقة المعتمدة على البيانات في أبكر مرحلة ممكنة في التدفق المتوازي. ويكون معيار الإرضاء: «افحص كلَّ تعليمة بالنسبة إلى سابقاتها وتجاهل ما يليها». وفي هذا المثال بالتحديد، تُخرِج خوارزمية الخدمة بأسبقية الوصول نتيجةً مثلى (أقصر تسلسل ممكن من المجموعات المتوازية). لكن قد يكون هناك تدفقات مدخلات أخرى تُنتج لها خوارزمية الخدمة بأسبقية الوصول مجموعات متوازية لا ترقى إلى المستوى الأمثل.

إذن، ما الفرق بين الخوارزميات الدقيقة والاستدلالية؟ في الخوارزميات الدقيقة، تقاس «الجودة» بتقييم تعقيد الوقت (أو المساحة) فيها. ولا يَلحق بالمخرجات مفاجآت أو عدم يقين. ولن يكون ثَم تباين في مخرجات خوارزميتين دقيقتين أو أكثر للمهمة الواحدة (مثل حل نظم المعادلات الجبرية أو فرز ملفات البيانات بترتيب تصاعدي، أو معالجة كشوف المرتبات، أو حساب العامل المشترك الأكبر لعددين صحيحين، أو غير ذلك)؛ ستختلفان (أو قد تختلفان) فقط في أدائهما أو الشكل الجمالي الخاص بهما. أما في حالة الخوارزميات الاستدلالية، فثمة الكثير من التفاصيل. بالتأكيد يمكن المقارنة بين الخوارزميات من حيث تعقيدات الوقت الخاصة بها. (خوارزمية الخدمة بأسبقية الوصول ذات تعقيد وقت $O(n^2)$ بالنسبة إلى تدفُّق مدخلات بالحجم n.) ولكن يمكن أيضًا مقارنتها من حيث المخرجات؛ لأن المخرجات قد تسبِّب مفاجآت في بعض الحالات. فربما نخرج بنتائجَ مختلفة من خوارزميتَي توازٍ أو برنامجَي شطرنج يستعملان مجموعات مختلفة من الأساليب الاستدلالية.

ومدى الاختلاف بينهما مسألة تجريبية. فلا بد من تنفيذ الخوارزميات في صورة برامج قابلة للتنفيذ، وإجراء التجارب على بيانات اختبارية متنوعة، ودراسة المخرجات، والتحقق من نقاط القوة ونقاط الضعف بناءً على التجارب. إذن، تنطوي الحوسبة الاستدلالية على إجراء التجارب.

الاستدلال والذكاء الاصطناعي

«الذكاء الاصطناعي» فرعٌ من علم الكمبيوتر ويهتم بالجوانب التنظيرية والتصميمية والتنفيذية لأدوات حوسبية تنفذ مهامَّ نربطها عادةً بالتفكير البشري: يمكن رؤية هذه الأدوات على أنها «تمتلك» ذكاءً اصطناعيًّا. وبذلك فهي تمثل جسرًا بين علم الكمبيوتر وعلم النفس. وأول مَن فكر في احتمالية تحقيق الذكاء الاصطناعي هو مهندس الكهرباء كلود شانون في أواخر أربعينيات القرن العشرين حينما تأمَّل فكرة برمجة كمبيوتر ليلعب الشطرنج. ومنذ ذلك الحين في الحقيقة، ظل شطرنج الكمبيوتر محلَّ اهتمام كبير في بحوث الذكاء الاصطناعي. لكن أكثر بيان مؤثِّر فيما عُرف بعد ذلك باسم الذكاء الاصطناعي (المصطلح ذاته صاغه أحدُ رواد المجال، وهو جون مكارثي في أواسط خمسينيات القرن العشرين) ورد في مقال مثير كتبه آلان تورنج (صاحب آلة تورنج) عام ١٩٥٠ حيث طرح السؤال التالي واقترح إجابة عنه: «ما الذي يعنيه الزعم بأن الكمبيوتر بإمكانه أن يفكر؟» وقد تضمنت إجابته إجراء تجربةٍ ما — «تجربة فكرية» — وفيها يطرح إنسان أسئلة على كيانين غير مرئيين عبر وسيلة اتصال «محايدة» (بحيث لا يستطيع السائل تخمين هوية المجيب عبر وسيلة الرد)، وكان أحد الكيانين إنسانًا والآخر جهاز كمبيوتر. وإذا لم يستطِع السائل أن يصيب في تخمين هوية الكمبيوتر باعتباره المجيب لأكثر من 40 إلى 50 بالمائة من الوقت، فيمكن اعتبار أن الكمبيوتر يُظهِر ذكاءً يشبه ذكاء الإنسان. عُرف هذا الاختبار باسم «اختبار تورنج» وكان بمثابة الغاية المنشودة في بحوث الذكاء الاصطناعي لسنوات.

مجال الذكاء الاصطناعي مجال شاسع، ويوجد في واقع الأمر أكثرُ من «نموذج فكري» يفضِّله باحثو الذكاء الاصطناعي. (أستخدم كلمة «نموذج» بالمعنى الذي يستخدمه فيلسوف العلم توماس كون.) ولكن كي نزيد من إيضاح نطاق الحوسبة الاستدلالية وقدراتها في هذا المقام، فلن أتناول غير «نموذج البحث الاستدلالي» في الذكاء الاصطناعي.

يهتم هذا النموذج بالكيانات الذكية — سواء الطبيعية أو الاصطناعية: أي، البشَر والآلات. كذلك يرتكز النموذج على فرضيتين ذكرهما مبتكِراه ألين نيويل وهيربرت سايمون بوضوح شديد وهما:

> **فرضية نظام الرموز المادية:** يمتلك نظام الرموز المادية الوسائل الضرورية والكافية للأفعال الذكية بوجه عام.
>
> **فرضية البحث الاستدلالي:** يحل نظام الرموز الفيزيائية المسائل عن طريق البحث على نحوٍ تدريجي وانتقائي (استدلالي) عبْر فضاء المسألة المكوَّن من البنيات الرمزية.

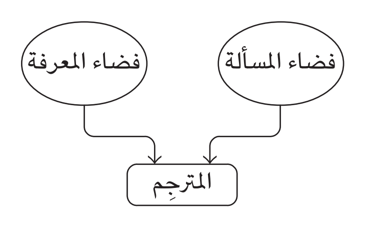
يقصد نيويل وسايمون بـ «نظام الرموز المادية» الأنظمةَ التي تعالج البنيات الرمزية وترتكز رغم ذلك على أساسٍ مادي — ما أسميته الأدوات الحوسبية المادية والحدية، غير أنها تتضمن الكائنات الطبيعية والاصطناعية تحت مظلتها على حد سواء.

يصوِّر الشكل ٦-١ صورةً عامة للغاية لكيان حل مسائل قائم على البحث الاستدلالي (سواء بشري أو اصطناعي). تُحَل المسألة أولًا بإنشاء تمثيل رمزي للمسألة في ذاكرة عاملة أو «فضاء مسألة». عادةً ما سيشير تمثيل المسألة إلى «الحالة الأولية» وهي التي يبدأ منها الكيان، و«الحالة الهدف» التي تمثِّل أحدَ حلول المسألة. إضافة إلى ذلك، يجب أن يكون فضاء المسألة قادرًا على تمثيل كل الحالات المحتملة التي يمكن الوصول إليها في محاولة الانتقال من الحالة الأولية إلى الحالة الهدف. وقد يكون فضاء المسألة هو ما يطلِق عليه علماء الرياضيات «فضاء الحالة». فضاء المسألة.

تحدُث الانتقالات من حالةٍ إلى أخرى بالاحتكام إلى محتويات «فضاء المعرفة» (محتويات الذاكرة الطويلة الأجل) لدى الكيان. كذلك تُنتقى العناصر من فضاء المعرفة هذا، وتُطبَّق على «الحالة الحالية» في فضاء المسألةِ، ما يؤدي إلى حالة جديدة. والجهة التي تفعل ذلك موضَّحة في الشكل ٦-١ باسم «المترجِم» (الذي سنتناوله بمزيد من التفصيل لاحقًا). تؤدي التطبيقات المتتالية لعناصر المعرفة ذات الصلة إلى قيام الكيان بالبحث عن حلٍّ عبر فضاء المسألة. وعملية البحث هذه تشكِّل عملية حوسبة. وتُحل المسألة عندما — بدءًا بالحالة الأولية — يؤدي تطبيق لسلسلة من عناصر المعرفة إلى الوصول إلى الحالة الهدف.

لكن وبما أن فضاء المسألة يمكن أن يكون كبيرًا للغاية، فإن البحث فيه لا يكون عشوائيًّا. بل يستخدم الكيان الأساليبَ الاستدلالية للتحكم في مقدار البحث، واستبعاد

أجزاء من فضاء البحث على أنها غير ضرورية، ومن ثَم الوصول إلى الحل في أسرع وقت ممكن.



شكل ٦-١: الهيكل العام لنظام البحث الاستدلالي.

الطرق الضعيفة والطرق القوية

إذن يكون صميم نموذج البحث الاستدلالي هو الأساليب الاستدلالية المتضمَّنة في فضاء المعرفة. قد تتراوح هذه الأساليب ما بين العامة جدًّا — بمعنى قابليتها لأن تنطبق على مجموعة كبيرة من مجالات المسائل — إلى الخاصة جدًّا — بمعنى أنها ذات صلة بمجالات مسائل معينة. يطلق على النوع الأول «الطرق الضعيفة»، ويطلق على النوع الثاني «الطرق القوية». وبوجه عام، عندما يكون مجال المسألة مفهومًا فهمًا رديئًا، تكون الطرق الضعيفة واعدة أكثر؛ وعندما يكون مجال المسألة معروفًا أو مفهومًا بمزيد من التفصيل، تكون الطرق القوية مناسبة أكثر.

من الطرق الضعيفة الفعالة (والتي قد أتينا على ذكرها عدة مرات) هي «فرِّق تسُد». وثمة طريقة أخرى تسمَّى «تحليل الوسائل والغايات»:

> بفرض وجود مسألة لها حالة حالية وحالة هدف، حدِّد الفرق بينهما. بعد ذلك اختزل الفرق عن طريق تطبيق «عامل» ذي صلة. لكن إذا لم يُستوفَ الشرط المسبق الضروري لتطبيق العامل، اختزل الفرق بين الحالة الحالية والشرط المسبق عن طريق تطبيق تحليل الوسائل والغايات على الحالة الحالية والشرط المسبق على نحو ذاتي الاستدعاء.

من الأمثلة التي يمكن أن تنطبق فيها استراتيجية فرِّق تسُد واستراتيجية تحليل الوسائل والغايات معًا هو مسألة تخطيط الطالب لبرنامج لنيل درجة علميةٍ ما. تقسم قاعدة فرِّق تسُد المسألةَ إلى مسائل فرعية بما يعادل كل سنة من سنوات برنامج الدرجة

العلمية. وتنقسم الحالة الهدف الأساسية (ولنقُل نيل الدرجة العلمية في تخصص معيَّن في عدد X من السنوات) إلى حالات «هدف فرعية» لكل سنة من السنوات التي عددها X. ولكل سنة، سيحدِّد الطالب الحالة الأولية لتلك السنة (المواد الدراسية التي دُرست قبل تلك السنة)، ويحاول أن يحدِّد المواد الدراسية المراد دراستها، التي تزيل الفرق بين الحالة الأولية والحالة الهدف لتلك السنة. ويتقلص البحث عن المواد الدراسية بتحديد المواد الإلزامية. لكن قد يتطلَّب بعض هذه المواد متطلباتٍ مسبقة أساسية. ومن ثَم تُطبق استراتيجية تحليل الوسائل والغايات لتقليل الفجوة بين الحالة الأولية والمتطلبات المسبقة الأساسية. وهكذا دواليك.

يرجى العلم أن استراتيجية تحليل الوسائل والغايات استراتيجيةٌ ذاتية الاستدعاء (ارجع إلى الفصل الثالث). إذن، ما وجه «الاستدلال» فيها؟ النقطة هنا هي أنه لا يوجد ما يضمن أن تنتهيَ استراتيجيةُ تحليل الوسائل والغايات بنجاح في مجال مسائل معين. على سبيل المثال، في ضوء حالة حالية وحالة هدف معينين، يمكن تطبيق عدة إجراءات لتقليص الفرق. وربما يحدد الإجراء الذي يقع عليه الاختيار الفرق بين النجاح والفشل.

وعادةً ما تمثِّل الطرق القوية «المعرفة الخبيرة» التي يحوزها المتخصصون في مجال المسألة عبْر التعليم الرسمي والتدريبات العملية والخبرة. ومن الأمثلة النموذجية على ذلك الأنظمة الحوسبية التي تحدِّد البنية الجزيئية للمواد الكيميائية أو تساعد المهندسين في مشاريع تصميماتهم. وغالبًا ما يعبَّر عن أساليب الاستدلال هذه في فضاء المعرفة باعتبارها قواعد (والتي تسمَّى «نواتج») باستخدام الصيغة التالية:

IF condition THEN action.

والمعنى أنه إذا تطابقت الحالة الحالية في فضاء المسألة مع جزء «الشرط» في القاعدة، فيمكن حينذاك تنفيذ «الإجراء» المقابل. ولنضرب مثالًا من مجال تصميم الدوائر الإلكترونية الرقمية (والذي ينفَّذ في إطار نظام استدلالي لأتمتة التصميمات) – والذي يشير هنا إلى أنه إذا كان الهدف من وحدة الدائرة هو تحويل الإشارة التسلسلية إلى إشارة متوازية، يجب استخدام أحد سجلات الإزاحة:

IF the goal of the circuit module is to convert a serial signal to a parallel one
THEN use a shift register.

من الممكن أن الحالة الحالية في فضاء المسألة تتطابق مع أجزاء الشرط لعدة قواعد:

IF condition1 THEN action1;
IF condition2 THEN action2.
....
IF condition*M* THEN action*M*.

في هذا الموقف، ربما يجب توجيه اختيار الإجراء المراد بأسلوب استدلالي ذي مستوًى أعلى (مثل اختيار أول قاعدة مطابقة). قد يتبيَّن أن هذا الاختيار كان خاطئًا كما سندرك لاحقًا في العملية الحوسبية، وفي هذه الحالة يجب أن «يرجع» النظام إلى حالة سابقة ويستكشف قاعدة أخرى.

ترجمة قواعد الاستدلال

لاحظ «المترجِم» في الشكل ٦-١. مهمة المترجِم تنفيذ خوارزمية دورية مناظرة لدورة التعليمات في الكمبيوتر المادي (ارجع إلى الفصل الخامس):

Match: Identify all the productions in the knowledge space the condition parts of which match the current state in the problem space. Collect these rules into a *conflict set.*

Select a preferred rule from the conflict set according to a selection heuristic.

Execute the action part of the preferred rule.
Goto Match.

بعيدًا عن عدم اليقين المصاحب لنموذج البحث الاستدلالي، فإن الفارق الكبير الآخر بين الخوارزميات (الدقيقة أو الاستدلالية) هو (كما ذكرنا من قبل) أن كل المعرفة اللازمة لتنفيذ الخوارزميات مدمجة في الخوارزمية نفسها. لكن في المقابل في نموذج البحث الاستدلالي، تقع المعرفة كلها تقريبًا في فضاء المعرفة (أو الذاكرة الطويلة الأجل). ويكمُن معظم تعقيد نموذج البحث الاستدلالي فيما يتمتع به فضاء المعرفة من ثراء.

الفصل السابع

التفكير الحوسبي

عقلية معينة

تتسم معظم العلوم في العصر الحديث — ولنقل بعد الحرب العالمية الثانية — بقدرٍ عالٍ من التخصص، بل الحصرية، بحيث يظل فهمُها المتعمق قاصرًا بدرجة كبيرة على المتخصصين في المجال، ونقصد بهم أوساطَ الممارسين لتلك العلوم. انظر على سبيل المثال إلى الفيزياء الحديثة المعنية بالجسيمات الأساسية. يُكشف عن تأثيرات تلك العلوم لعموم الناس في أفضل الأحوال وعندما يكون ذلك مناسبًا من خلال عواقبها وآثارها التكنولوجية.

لكن يوجد بعض العلوم التي تمس خيال غير المتخصصين بفعل الطبيعة الجذابة لأفكارها المحورية. ومن الأمثلة على ذلك في مجال العلوم الطبيعية نظرية التطور. فقد امتدَّت أذرع تأثير تلك النظرية إلى علم الاجتماع وعلم النفس والاقتصاد وحتى علم الكمبيوتر، وهي مجالات فكرية لا علاقة لها بالجينات أو الانتقاء الطبيعي.

ومن بين العلوم الاصطناعية، يُظهِر علم الكمبيوتر سمةً مشابهة لذلك. وأنا لا أشير إلى الأدوات التكنولوجية الواسعة الانتشار التي غزت العالم الاجتماعي. بل أشير إلى ظهور «عقلية» معينة.

عبَّر أحد رواد الذكاء الاصطناعي، وهو سيمور بابيرت، عن هذه العقلية — أو على الأقل بشائر ظهورها — بأسلوب حماسي وبليغ في كتاب من تأليفه بعنوان «العواصف الذهنية» (١٩٨٠). أعلن بابيرت أن هدفه من هذا الكتاب أن يناقش ويذكر كيف أن الكمبيوتر يمكن أن يوفِّر للإنسان طرقًا جديدة للتعلم والتفكير، ليس باعتباره أداةً عملية ومفيدة فحسب، بل بطرق جوهرية ومفاهيمية أكثر. وستتيح هذه التأثيرات أنماطًا للتفكير حتى عندما لا يكون الشخص المفكِّر على اتصال مباشر مع الجهاز المادي. وفي رأي بابيرت، يبشر الكمبيوتر بأنه سيكون بمثابة «ناقل محتمل للأفكار العظيمة وبذور

التغيير الثقافي». وقد وعد بأن يتناول كتابه كيف يمكن للكمبيوتر أن يساعد البشرَ مساعدةً مثمرة في تجاوز الحدود التقليدية التي تفصِل بين المعرفة الموضوعية والمعرفة الذاتية، وبين الإنسانيات والعلوم.

ما تناوله بابيرت كان رؤية — ربما كانت مثالية — تجاوزت التأثيرَ المادي البحت لأجهزة الكمبيوتر والحوسبة في شئون العالم. وقد ظهرت هذه الرؤية الأخيرة منذ بدايات الحوسبة التلقائية التي نشأت في عصر تشارلز بابيج وآدا كونتيسة لوفليس في أواسط القرن التاسع عشر. بل إن رؤية بابيرت كانت تلقينًا لعقليةٍ ستوجِّه وتشكِّل وتؤثِّر في طرق تفكير الشخص في أوجه العالم وإدراكه واستجابته لها — سواء عالَم المرء الداخلي أو العالَم الخارجي — التي تبدو للوهلة الأولى أنها لا علاقة لها بالحوسبة، ربما من حيث القياس أو الاستعارة أو الخيال.

وبعد ما يزيد على ربع قرن من صدور كتاب بابيرت، منحت عالِمة الكمبيوتر جانيت وينج اسمًا لهذه العقلية وهو «التفكير الحوسبي». لكن ربما كانت رؤية وينج واقعيةً أكثر من رؤية بابيرت. ففي عام ٢٠٠٨، كتبت تقول إن التفكير الحوسبي ينطوي على مناهجَ لأنشطة مثل حل المسائل والتصميم وإدراك السلوك الذكي الذي يعتمد على المفاهيم الأساسية للحوسبة. لكن لا يمكن أن يكون التفكير الحوسبي جزيرةً قائمةً بذاتها. ففي مجال حل المسائل، سيكون أقرب إلى التفكير الرياضي؛ وفي مجال التصميم، سيتشارك سمات مع العقلية الهندسية؛ وفي فهم الأنظمة الذكية (بما فيها العقل بالطبع)، فقد يجد أرضًا مشتركة مع التفكير العلمي.

ومثل بابيرت، فصلت وينج بين عقلية التفكير الحوسبي عن الكمبيوتر المادي ذاته؛ أي إن المرء بإمكانه التفكير حوسبيًّا من دون وجود الكمبيوتر.

لكن ما الذي تنطوي عليه عقلية التفكير الحوسبي هذه؟ سنذكر بعضَ الأمثلة لاحقًا، لكن قبل ذلك دعنا نتتبَّع تفسير الباحث في مجال الذكاء الاصطناعي بول روزنبلوم لمفهوم التفكير الحوسبي من منطلق نوعين من العلاقات؛ أحدهما هو «التفاعل»، وقد سبق أن طرحنا هذا المفهوم (ارجع إلى الفصل الثاني)، ويعني — من منظور روزنبلوم — «الأفعال أو التأثيرات أو الآثار المتبادلة» بين كيانين. لكن قد يشير التفاعل إلى تأثيرٍ أحادي الاتجاه من جانب نظام واحد A على نظام آخر B (صوَّره روزنبلوم رمزيًّا بالآتي: $A \longrightarrow B$ أو $B \longleftarrow A$)، وكذا إلى تأثيرٍ ثنائي الاتجاه أو متبادل (ويُرمز له بالآتي: $A \longleftarrow \longrightarrow B$). ويقصد روزنبلوم بمصطلح «التنفيذ»، وهو النوع الآخر من العلاقات، وضعَ النظام A

الذي في مستوى تجريدٍ أعلى من حيث تفاعل العمليات داخل النظام B الذي عند مستوى تجريدٍ أقل (ويُرمز لذلك بالآتي: A/B). ومن حالات التنفيذ الخاصة «المحاكاة»: النظام B يحاكي النظام A (ويُرمز إليه A/B) عندما يقلِّد النظام B سلوك النظام A أو يحاكيه.

باستخدام هاتين العلاقتين، كما يوضح روزنبلوم، يتجلَّى أبسط تمثيل للتفكير الحوسبي عندما تؤثِّر الأداة الحوسبية C في سلوك إنسان H‏ (H): C → H. ثم يذهب روزنبلووم إلى ما هو أبعدَ من ذلك. فبدلًا من مجرد وجود كائن بشري H، هبْ أننا ننظر إلى إنسانٍ يحاكي أداةً حوسبية C: C/H. في هذه الحالة، تتخذ العلاقة الشكلَ C → C/H، ما يعني أن الأدوات الحوسبية تؤثِّر في البشر الذين يحاكون سلوكَ هذه الأدوات. أو يمكننا الذهاب إلى ما هو أبعد أكثر وأكثر: هبْ أن إنسانًا H يحاكي أداةً حوسبية C محاكاةً عقلية، وهذه الأداة نفسها نفَّذت أو تحاكي مجالًا من العالم الحقيقي D: D/C/H. على سبيل المثال، لنفترض أن D تعبِّر عن سلوك بشري. إذن D/C تعني استخدام الكمبيوتر لمحاكاة سلوك الإنسان أو تجسيده. كذلك D/C/H تعني أن الإنسان يحاكي تجسيدَ الكمبيوتر هذا لسلوك الإنسان محاكاةً ذهنية. وهذا يؤدي إلى التفسير التالي للتفكير الحوسبي: C → D/C/H.

قد يكون هناك تفسيراتٌ أدق، ولكن هذه التفسيرات من حيث التفاعل والتنفيذ/ المحاكاة كافية لتوضيح النطاق العام للتفكير الحوسبي.

التفكير الحوسبي باعتباره مهاراتٍ ذهنية

أوضحُ تأثيرٍ يمكن أن تحدِثه الحوسبة في الإنسان هو أن تكون مصدرًا للمهارات الذهنية؛ أي تكون مجموعة من أدوات التحليل وحل المسائل التي يمكن أن يطبِّقها الإنسان في سياق حياته بغض النظر عن وجود أجهزة الكمبيوتر الفعلية من عدمه. هذا ما كانت تقصده جانيت وينج. تحديدًا، إنها اتخذت التجريد باعتباره «الجوهر» — الأساس — للتفكير الحوسبي. لكن بما أن التجريد دون شك مفهوم حوسبي أساسي (كما رأينا بين صفحات هذا الكتاب)، فإن علم الكمبيوتر يقدِّم الكثير من الأفكار والمفاهيم التي يمكن أن يستوعبها الإنسان ويدمجها في مجموعة أدوات التفكير لديه. أقصد بذلك الطرق الاستدلالية، سواء الضعيفة أو القوية؛ وفكرة الوصول لحلٍّ مُرضٍ بدلًا من التحسين باعتباره غايةً واقعية لعملية اتخاذ القرار؛ والتفكير بطريقة خوارزمية وفهم متى يكون هذا المسار مناسبًا لحل المسألة، وهل هو مناسب أم لا؛ وشروط ومعمارية المعالجة المتوازية باعتبارها وسيلةً

للتعامل مع المساعي المتعددة المهام؛ وأسلوب حل المسائل بطريقة «تنازلية» (بحيث نبدأ بالهدف والحالة الأولية للمسألة، ثم نقسم الهدف إلى أهداف فرعية أبسط، ونقسم الأهداف الفرعية إلى المزيد من الأهداف الفرعية الأبسط، وهكذا دواليك)، أو بطريقة «تصاعدية» (بحيث نبدأ بالهدف وأدنى لبنات بنائه في المستوى، ثم ننشئ حلًّا بتنظيم لبنات البناء لصناعةِ لبناتِ بناءٍ أكبرَ، وهكذا). لكن المهم هو أن اكتساب أدوات التفكير هذه يتطلب مستوًى معينًا من إتقان مفاهيم علم الكمبيوتر. ومن وجهة نظر وينج، ينطوي هذا على إدخال التفكير الحوسبي ضمن المناهج التعليمية منذ المراحل الأولى.

لكن التفكير الحوسبي ينطوي على ما هو أكثر من المهارات التحليلية وتلك الخاصة بحل المسائل. فهو يشمل طريقةً «للتخيل»، عن طريق رؤية التشابهات وصياغة الاستعارات. هذا المزيج من المهارات التقنية والخيال هو ما كان يفكِّر فيه بابيرت — على ما أعتقد — والذي يوفِّر الثراء الكامل لعقلية التفكير الحوسبي. سنتناول الآن بعضَ مجالات البحث الفكري والعلمي التي أثبتت هذه العقلية فاعليتها فيها.

التفكير في العقل بطريقة حوسبية

أحد أقوى مظاهر هذه العقلية بالتأكيد — وإن كان مثيرًا للجدل — هو التفكير في التفكير: بمعنى تأثير علم الكمبيوتر في علم النفس المعرفي. بطرح سؤال تورنج الشهير — هل بإمكان أجهزة الكمبيوتر التفكير (الذي هو أساس الذكاء الاصطناعي)؟ — بنحو عكسي، يدرُس علماء علم النفس المعرفي السؤال التالي: هل التفكير عملية حوسبية؟

تعود الإجابة عن هذا السؤال إلى العمل الرائد الذي قدَّمه ألين نيويل وهيربرت سايمون في أواخر خمسينيات القرن العشرين، حينما وضعا نظريةَ معالجة المعلومات الخاصة بعملية حل المسائل البشرية، والتي تمزج المنطق بمسائل حوسبية مثل الأساليب الاستدلالية ومستويات التجريد والبنيات الرمزية. وقد أدَّى ذلك في وقتٍ لاحق جدًّا إلى بناء نماذج من «المعمارية المعرفية»، والتي كان أبرزها على يد باحثِين أمثال عالم النفس جون أندرسون، وعلماء الكمبيوتر ألين نيويل، وجون لايرد، وبول روزنبلوم. تأثَّرت سلسلة النماذج التي وضعها أندرسون — وتسمَّى بوجه عام ACT (التحكم التكيفي للتفكير) — والنماذج التي وضعها نيويل وآخرون — وتسمَّى SOAR — تأثُّرًا كبيرًا بالمبادئ الأساسية لمعمارية الكمبيوتر الداخلية (ارجع إلى الفصل الخامس). في هذه النماذج، تُستكشف معمارية الإدراك من حيث التسلسلات الهرمية للذاكرة التي تحمل البنيات الرمزية التي

تمثِّل جوانبَ العالم، واستخدام البنيات الرمزية ومعالجتها عن طريق عملياتٍ تشبه دورة تفسير التعليمات. خضعت هذه النماذج المعمارية لدراسات مكثَّفة على الصعيدين النظري والتجريبي باعتبارها نظريات محتملة للعقل المفكر على مستوًى معين من التجريد. ويبدأ نوعٌ آخر من نماذج العقل المتأثرة بالحوسبة بمبادئ المعالجة المتوازية والحوسبة الموزعة، ويرى العقل على أنه «مجتمع» من الوحدات المعرفية الموزَّعة والمتواصلة والمتفاعلة بعضها مع بعض. ومن المؤيدين المؤثرين لهذا النوع من النمذجة العقلية رائدُ الذكاء الاصطناعي مارفن مينسكي. وبالنسبة إلى عالمة العلوم المعرفية والفيلسوفة مارجريت بودن، فقد وضعت عنوانًا لكتابها الرائع الذي يعرض لتاريخ العلوم المعرفية هو «العقل باعتباره آلة» (٢٠٠٦)، وطبقًا لروايتها فيه، فإن «العقل جهاز حوسبي».

الدماغ الحوسبي

يعود تاريخ تمثيل أو نمذجة البنية العصبية للدماغ باعتبارها نظامًا حوسبيًّا، وكذلك اعتبار الأدوات الحوسبية شبكاتٍ من الكيانات ذات مستوى تجريدٍ عالٍ وتشبه الخلايا العصبية إلى العمل الرائد لعالِم الرياضيات وران بيتس وعالِم الفسيولوجيا العصبية وارن مكولوك، والعالِم النابض بالحيوية جون فون نيومان في أربعينيات القرن العشرين. وعلى مدار الستين سنة التالية، تطوَّر نموذج علمي يسمى «الترابطية». في هذا النهج، يعبَّر عن عقلية التفكير الحوسبي على نحوٍ أكثر تحديدًا من خلال تصميم شبكات ذات درجة ترابط عالية فيما بينها (ومن هنا ظهر مصطلح «الترابطية») تتكوَّن من عناصر حوسبية بسيطة للغاية تعمل جميعها لنمذجة سلوك العمليات الأساسية للدماغ، التي هي عبارة عن لبنات بناء في عمليات معرفية ذات مستوًى أعلى (مثل رصد الدلائل أو التعرُّف على الأنماط في العمليات البصرية). وتقع المعماريات الترابطية للدماغ عند مستوى تجريد أقل من المعماريات المعرفية القائمة على معالجة الرموز التي ذكرناها في القسم السابق.

ظهور العلوم المعرفية

معماريات العقل المعرفية القائمة على معالجة الرموز ونماذج الدماغ الترابطية، هما من الطرق التي أثَّرت بها الأدوات الحوسبية ومبادئ علم الكمبيوتر في تشكيل وظهور مجال العلوم المعرفية المتعدد التخصصات والجديد نسبيًّا. وقد وجب التنويه إلى أن علماء العلوم

المعرفية لم يتخذوا كلهم — مثل عالِم النفس جيروم برونر — الحوسبة باعتبارها عنصرًا جوهريًّا في المعرفة. وعلى الرغم من ذلك، فإن فكرة فهْم أنشطة مثل التفكير والتذكر، والتخطيط وحل المشكلات واتخاذ القرار والإدراك، وصياغة المفاهيم والفهم عن طريق إنشاء نماذج حوسبية ووضع فرضيات قائمة على الحوسبة تُعدُّ فكرة فاتنة؛ ولا سيما أن رؤية علم الكمبيوتر باعتباره علمَ المعالجة التلقائية للرموز شكَّلت «حافزًا» قويًّا في ظهور العلوم المعرفية نفسها. ويقوم جوهر تأريخ مارجريت بودن للعلوم المعرفية — المذكور في القسم السابق — على تطوير الحوسبة التلقائية.

فهم الإبداع البشري

الموضوع الجذاب للإبداع — بدايةً من النوع الاستثنائي والأصيل تاريخيًّا وحتى الصنف الشخصي اليومي — موضوعٌ متشعب استقطب اهتمامًا مهنيًّا من علماء النفس والمحللين النفسيين والفلاسفة والتربويين وعلماء الجمال والمنظرين الفنيين ومنظري التصميم والمؤرخين الفكريين وكتَّاب السير الذاتية؛ فضلًا عن المبدعين أنفسهم الأكثر استبطانًا لذواتهم (العلماء والمخترعين والشعراء والكُتاب والموسيقيين والفنانين وغيرهم). إن نطاق مناهج الإبداع ونماذجه ونظرياته كبيرٌ للغاية، ومن أهم الأسباب في ذلك التعريفات العديدة له.

لكن على الأقل لجأت جماعةٌ واحدة من الباحثين في الإبداع إلى التفكير الحوسبي باعتباره «طريقة عمل». اقترح هؤلاء نظرياتٍ ونماذجَ حوسبية لعملية الإبداع تعتمد اعتمادًا كبيرًا على مبادئ الحوسبة الاستدلالية، وعلى تمثيل المعرفة في صورة بنيات رمزية معقَّدة (تسمَّى مخططات)، وتعتمد كذلك على مبادئ التجريد. هنا أيضًا نجد التأثير الدامغ للتفكير الحوسبي؛ حيث مهَّد أرضًا مشتركة لتحليل الإبداع العلمي والتكنولوجي والفني والأدبي والموسيقي: أي، تزاوج العديد من الثقافات كما كان بابيرت يأمُل.

على سبيل المثال، طبَّق الباحث الأدبي مارك تورنر المبادئَ الحوسبية على مسألة فهم عملية التأليف الأدبي، تمامًا كما سعى فيلسوف العلم وعالِم العلوم المعرفية بول ثايجارد إلى تفسير الثورات العلمية باستخدام النماذج الحوسبية، وصاغ مؤلِّف الكتاب الذي بين أيديكم — الذي هو عالِم كمبيوتر وباحث في مجال الإبداع — تفسيرًا حوسبيًّا لتصميم وابتكار الأفكار والأدوات التكنولوجية في العلوم الاصطناعية. لقد كانت عقلية التفكير الحوسبي بمثابة الغراء الذي يربط بين هذه الثقافات الفكرية والإبداعية المختلفة في ثقافة

واحدة. وفي العديد من هذه الدراسات الحوسبية الخاصة بالإبداع، قدَّم علم الكمبيوتر دقةً في التفكير للتعبير عن المفاهيم المتعلقة بالإبداع التي كانت غائبة من قبل.

لنضرب مثالًا، اعتبر الكاتب آرثر كيستلر في كتابه البالغ الأهمية «عمليةَ الإبداع» (١٩٦٤) عملية تسمَّى «الارتباط الثنائي» بمنزلة آليةٍ تتأثَّر بها الأعمال الإبداعية. يقصد كيستلر بمصطلح الارتباط الثنائي الجمع بين مفهومين غير مترابطين أو أكثر ودمجهما، ما يؤدي إلى منتجٍ أصلي. ومع ذلك، ظلَّت الطريقةُ الدقيقة التي يحدُث بها الارتباط الثنائي مبهمة. وقد قدَّم التفكير الحوسبي لبعض الباحثين في مجال الإبداع (مثل مارك تورنر وهذا الكاتب) تفسيراتٍ لارتباطات ثنائية معينة باللغة الدقيقة لعلم الكمبيوتر.

فهم المعالجة الجزيئية للمعلومات

لقد اكتشف جيمس واتسون، وفرانسيس كريك، بنيةَ جزيء الحمض النووي عام ١٩٥٣. وعلى إثر ذلك، بدأ علم الأحياء الجزيئي. ومن بين ما يهتم به هذا العلم فهْم واكتشاف آليات كآليات تضاعف الحمض النووي ونسخ جزء من الحمض النووي إلى الحمض النووي الريبوزي، وترجمة الحمض النووي الريبوزي إلى بروتين — وهذه من العمليات البيولوجية الأساسية. ومن هنا، دخلت إلى الوعي الأحيائي فكرةُ تصوير الجزيئات بأنها ناقلات معلومات. فبدأ منظِّرو علم الأحياء المتأثرون بالأفكار الحوسبية في نمذجة العمليات الوراثية من منظورٍ حوسبي (والذي أدَّى مصادفةً إلى ابتكار خوارزميات بناءً على مفاهيمَ وراثية). ومن ثَم شكَّل التفكير الحوسبي ما كان يسمَّى «معالجة المعلومات الأحيائية» أو ما يُسمى باللغة المعاصرة «المعلوماتية الأحيائية».

الخاتمة: هل يندرج علم الكمبيوتر ضمن العلوم الشاملة؟

عبْر هذا الكتاب، كانت الفرضية المطروحة هي أن علم الكمبيوتر يندرج ضمن العلوم الاصطناعية؛ ذلك أنه يرتكز على أدوات معالجة رمزية (أو حوسبية)، وأنه من العلوم التي تهتم بالكيفية التي من المفترض أن تكون عليها الأشياء وليس بتلك التي هي عليها بالفعل، وأنه يجب أخذُ أهداف منشئي الأدوات الحوسبية (مصمِّمي الخوارزميات، والمبرمجين، ومهندسي البرمجيات، واختصاصيي معمارية الكمبيوتر، واختصاصيي نُظم المعلومات) في الاعتبار عند فهْم طبيعة هذا العلم. في كل هذه النواحي، الفرق بينه وبين العلوم الطبيعية واضح.

لكن رأينا في الفصل الأخير أن التفكير الحوسبي يعمل بمثابة جسرٍ بين عالم الأدوات الحوسبية والعالم الطبيعي، وعلى وجه التحديد، عالم الجزيئات الأحيائية والإدراك البشري والعمليات العصبية. إذن، أيمكن ألَّا توفِّر الحوسبةُ عقليةً فقط، وإنما تشمل أيضًا — بشكل أكثرَ دهاءً — ظاهرةَ العالمين الطبيعي والاصطناعي؟ وأن علم الكمبيوتر يندرج ضمن العلوم «الشاملة»؟

في السنوات الأخيرة، فكَّر بعض علماء الكمبيوتر في هذا الاتجاه تحديدًا. وعليه حاجج بيتر دينينج بأنه ما عاد ينبغي النظر إلى الحوسبة باعتبارها علمًا اصطناعيًّا؛ حيث إن عمليات معالجة المعلومات توجد بغزارة في الطبيعة. وقد أكَّد دينينج وعالم كمبيوتر آخر وهو بيتر فريمان أنه في العقود القليلة الماضية تحوَّل تركيز الانتباه (الخاص ببعض علماء الكمبيوتر) من الأدوات الحوسبية إلى عمليات معالجة المعلومات في حد ذاتها — بما في ذلك عمليات معالجة المعلومات الطبيعية.

من ثَم، ومن وجهة نظر دينينج وفريمان وعالم كمبيوتر آخر وهو ريتشارد سنودجراس، تندرج الحوسبة ضمن العلوم «الطبيعية»؛ حيث إن علماء الكمبيوتر معنيون باكتشاف «الكيفية التي عليها الأشياء» (في الدماغ وفي الخلايا الحية وحتى في عالم الأدوات الحوسبية) بقدرِ ما هم معنيون بتوضيح الكيفية التي من المفترض أن تكون عليها الأشياء. يشير هذا الرأي إلى أن الأدوات الحوسبية تنتمي إلى الفئة الوجودية نفسها التي تنتمي إليها الكيانات الطبيعية؛ أو أنه لا يوجد فرقٌ بين ما هو طبيعي وما هو اصطناعي. بل إن سنودجراس في واقع الأمر ابتكر كلمةً لوصف الجانب الطبيعي في علم الكمبيوتر، وهي Ergalics وهي مشتقة من الجذر اليوناني ergon وتعني «العمل».

وفي اتفاق واسع النطاق من جانبه مع سنودجراس، ورغبةً منه أيضًا في تجنُّب ابتكار ألفاظ جديدة، ضمَّ بول روزنبلوم «علوم» الكمبيوتر إلى جانب العلوم الفيزيائية والحياتية والاجتماعية وقال بأنها «رابع المجالات العلمية العظيمة».

تفرُّد علم الكمبيوتر باعتباره نموذجًا معرفيًّا قائمًا بذاته كان موضوعًا ثابتًا في هذا الكتاب، ومن ثَم تتوافق أطروحة روزنبلوم وتناوُل الكتاب. السؤال هو ما إذا كان ينبغي التمييز بين دراسة عمليات «معالجة المعلومات الطبيعية» وعمليات «المعالجة الرمزية الاصطناعية». هنا، يبدو أن الفرق بين المعلومات والرموز له مسوِّغاته. في المجالات الطبيعية، لا تمثل الكيانات شيئًا غير نفسها. فالكيانات، كالخلايا العصبية أو النوكليوتيدات التي هي لبنة بناء الحمض النووي، أو الأحماض الأمينية التي تشكِّل البروتينات، لا تمثل أي شيء غير نفسها. ومن ثَم، أجد أنه من الصعب الإشارة إلى عمليات معالجة الحمض النووي على أنها عمليات معالجة «رمزية»، على الرغم من أن الإشارة إلى هذه الكيانات على أنها ناقلات للمعلومات غير المرجعية تبدو صحيحة.

من المنظور الوجودي، أعتقد أنه ينبغي التمييز بين علم الكمبيوتر باعتباره علمًا اصطناعيًّا وعلم الكمبيوتر باعتباره علمًا طبيعيًّا. ففي الأول، الوساطة البشرية (في شكل الأهداف والغاية، والوصول إلى المعرفة، وإحداث تأثير) جزءٌ من العلم. أما في الثاني، فلا يخفى على أحد غيابُ هذه الوساطة. فالنموذجان المعرفيان مختلفان جوهريًّا بعضهما عن بعض.

مهما كان الأمر وبغض النظر عن أي اختلاف وجودي محتمَل مثل هذا، فإن ما قدَّمه لنا علم الكمبيوتر — كما حاونا التوضيح في الفصول السابقة — هو طريقة مميزة بنحو ملحوظ لإدراك مجموعة هائلة للغاية من المسائل — تندرج تحت مجالات طبيعية

واجتماعية وثقافية وتكنولوجية واقتصادية — والتفكير فيها وحلها. هذا بالتأكيد هو إسهامه العلمي الأكثر «أصالة» في العالم الحديث.

قراءات إضافية

قد يرغب القارئ في دراسة الموضوعات التي تناولتها مختلف فصول الكتاب بمزيد من التعمُّق. وفيما يلي قائمة تضم مجموعة من الأعمال الكلاسيكية والمؤثرة تاريخيًّا (التي لا تزال مقروءة بصورة بارزة)، وكذلك مجموعة من النصوص الأحدث عهدًا؛ وهي مجموعة من المقالات والأعمال التاريخية المخصَّصة لقطاع عريض من القراء، وكذلك مقالات متخصصة أكثر إلى حدٍّ ما.

المقدمة

S. Dasgupta (2014). *It Began with Babbage: The Genesis of Computer Science*. New York: Oxford University Press: esp. chapter 15.

الفصل الأول: «مقومات» الحوسبة

S. Dasgupta (2014). *It Began with Babbage: The Genesis of Computer Science.* New York: Oxford University Press: chapters 1 & 2.

L. Floridi (2010). *Information: A Very Short Introduction.* Oxford: Oxford University Press.

D. Ince (2011). *Computer: A Very Short Introduction.* Oxford: Oxford University Press.

D.E. Knuth (1996). 'Algorithms, Programs and Computer Science' (originally published in *1966*). *Selected Papers in Computer Science.* Stanford, CA: Center for the Study of Language and Information.

A. Newell, A.J. Perlis, & H.A. Simon (1967). 'What is Computer Science?' *Science*, 157, 1373–4.

A. Newell & H.A. Simon (1976). 'Computer Science as Empirical Inquiry: Symbols and Search', *Communications of the ACM*, 19, 113–26.

P.S. Rosenbloom (2010). *On Computing: The Fourth Great Scientific Domain*. Cambridge, MA: MIT Press.

الفصل الثاني: الأدوات الحوسبية

C.G. Bell, J.C. Mudge, & J.E. McNamara (1978). 'Seven Views of Computer Systems', pp. 1–26, in C.G. Bell, J.C. Mudge, & J.E. McNamara (ed.). *Computer Engineering: A DEC View of Hardware Systems Design.* Bedford, MA: Digital Press.

C.G. Bell, D.P. Siweorek, & A. Newell (1982). *Computer Structures: Principles and Examples.* New York: McGraw-Hill: esp. chapter 2.

S. Dasgupta (2014). *It Began with Babbage: The Genesis of Computer Science.* New York: Oxford University Press: esp. prologue and chapter 4.

J. Copeland (ed.) (2004). *The Essential Turing.* Oxford: Oxford University Press.

J. Copeland (2004). 'Computing', pp. 3–18, in L. Floridi (ed.). *Philosophy of Computing and Information.* Oxford: Blackwell.

E.W. Dijkstra (1968). 'The Structure of "THE" Multiprogramming System', *Communications of the ACM*, 11, 341–6.

E.W. Dijkstra (1971). 'Hierarchical Ordering of Sequential Processes', *Acta Informatica*, 1, 115–38.

D. Ince (2011). *The Computer: A Very Short Introduction.* Oxford: Oxford University Press.

H.H. Pattee (ed.) (1973). *Hierarchy Theory: The Challenge of Complex Systems.* New York: Braziller.

H.A. Simon (1996). *The Sciences of the Artificial* (3rd edn). Cambridge, MA: MIT Press.

A.S. Tanenbaum & H. Bos (2014). *Modern Operating Systems* (4th edn). Englewood Cliffs, NJ: Prentice-Hall.

الفصل الثالث: التفكير الخوارزمي

J. Copeland (2004). 'Computing', pp. 3–18, in L. Floridi (ed.). *Philosophy of Computing and Information.* Oxford: Blackwell.

E.W. Dijkstra (1965). 'Programming Considered as a Human Activity', *Proceedings of the 1965 IFIP Congress.* Amsterdam: North-Holland, pp. 213–17.

D.E. Knuth (1996). 'Algorithms', pp. 59–86, in D.E. Knuth. *Selected Papers on Computer Science*. Stanford, CA: Center for the Study of Language and Information.

D.E. Knuth (1997). *The Art of Computer Programming. Volume 1. Fundamental Algorithms* (3rd edn). Reading, MA: Addison-Wesley.

D.E. Knuth (2001). 'Aesthetics', pp. 91–138, in D.E. Knuth. *Things a Computer Scientist Rarely Talks About.* Stanford, CA: Center for the Study of Language and Information.

R. Sedgewick & K. Wayne (2011). *Algorithms* (4th edn). Reading, MA: Addison-Wesley.

الفصل الرابع: فن البرمجة وعلمها وهندستها

F.P. Brooks, Jr (1975). *The Mythical Man-Month: Essays on Software Engineering*. Reading, MA: Addison-Wesley.

P. Freeman (1987). *Software Perspectives*. Reading, MA: Addison-Wesley.

C.A.R. Hoare (1985). *The Mathematics of Programming.* Oxford: Clarendon Press.

C.A.R. Hoare (2006). 'The Ideal of Program Correctness'. http://www.bcs.org/upload/pdf/correctness.pdf. Retrieved 28 May 2014.

D.E. Knuth (1992). *Literate Programming*. Stanford, CA: Center for the Study of Language and Information. See esp. 'Computer Programming as Art', pp. 1–16.

D.E. Knuth (2001). *Things a Computer Scientist Rarely Talks About.* Stanford, CA: Center for the Study of Language and Information. See esp. 'Aesthetics', pp. 91–138.

I. Sommerville (2010). *Software Engineering* (9th edn). Reading, MA: Addison-Wesley.

M.V. Wilkes (1995). *Computing Perspectives.* San Francisco: Morgan Kauffman. Esp. 'Software and the Programmer', pp. 87–92; 'From FORTRAN and ALGOL to Object-Oriented Languages', pp. 93–101.

N. Wirth (1973). *Systematic Programming: An Introduction*. Englewood Cliffs, NJ: Prentice Hall.

الفصل الخامس: مجال معمارية الكمبيوتر

G.S. Almasi & A. Gottlieb (1989). *Highly Parallel Computing.* New York: The Benjamin Cummings Publishing Company.

C.G. Bell, J.C. Mudge, & J.E. McNamara (ed.) (1978). *Computer Engineering: A DEC View of Hardware Systems Design.* Bedford, MA: Digital Press.

C.G. Bell, D.P. Sieweorek, & A. Newell (1982). *Computer Structures: Principles and Examples.* New York: McGraw-Hill.

S. Dasgupta (2014). *It Began with Babbage: The Genesis of Computer Science.* New York: Oxford University Press.

S. Habib (ed.) (1988). *Microprogramming and Firmware Engineering Methods.* New York: Van Nostrand Reinhold.

C. Hamachar, Z. Vranesic, & S. Zaky (2011). *Computer Organization and Embedded Systems* (5th edn). New York: McGraw-Hill.

K. Hwang & F.A. Briggs (1984). *Computer Architecture and Parallel Processing.* New York: McGraw-Hill.

D.E. Ince (2011). *The Computer: A Very Short Introduction.* Oxford: Oxford University Press.

D.A. Patterson & J.L. Henessy (2011). *Computer Architecture: A Quantitative Approach* (5th edn). Burlington, MA: Morgan Kaufmann.

A.S. Tanenbaum (2011). *Structured Computer Organization* (6th edn). Englewood Cliffs, NJ: Prentice Hall.

الفصل السادس: الحوسبة الاستدلالية

D.R. Hofstadter (1999). *Gödel, Escher, Bach: An Eternal Golden Braid* (20th anniversary edn). New York: Basic Books.

A. Newell & H.A. Simon (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice Hall.

A. Newell & H.A. Simon (1976). 'Computer Science as Empirical Inquiry: Symbols and Search', *Communications of the ACM*, 19, 113–26.

J. Pearl (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, MA: Addison-Wesley.

G. Polya & J.H. Conway (2014). *How to Solve It: A New Aspect of Mathematical Method.* Princeton, NJ: Princeton University Press. (Originally published in 1949).

D.L. Poole & A.K. Mackworth (2010). *Artificial Intelligence: Foundations of Computational Agents.* Cambridge: Cambridge University Press.

S. Russell & P. Norvig (2014). *Artificial Intelligence: A Modern Approach* (3rd edn). New Delhi: Dorling Kinderseley/Pearson.

H.A. Simon (1995). 'Artificial Intelligence: An Empirical Science', *Artificial Intelligence* 77, 1, 95–127.

H.A. Simon (1996). *The Sciences of the Artificial* (3rd edn). Cambridge, MA: MIT Press.

الفصل السابع: التفكير الحوسبي

M.A. Boden (2006). *Minds as Machines. Volume 1.* Oxford: Clarendon Press.

S. Dasgupta (1994). *Creativity in Invention and Design: Computational and Cognitive Explorations of Technological Originality.* New York: Cambridge University Press.

S. Papert (1980). *Mindstorms*. New York: Basic Books.

P.S. Rosenbloom (2013). *On Computing: The Fourth Great Scientific Domain*. Cambridge, MA: MIT Press.

J. Searle (1984). *Minds, Brains and Science*. Cambridge, MA: Harvard University Press.

J. Setubal & J. Meidinis (1997). *Introduction to Computational Molecular Biology.* Pacific Grove, CA: Brooks/Cole Publishing Company.

C.A. Stewart (ed.) (2004). '*Bioinformatics:* Transforming Biomedical Research and Medical Care' [Special Section on Bioinformatics], *Communications of the ACM*, 47/11, 30–72.

P.R. Thagard (1988). *Computational Philosophy of Science*. Cambridge, MA: MIT Press.

P.R. Thagard (1992). *Conceptual Revolutions*. Princeton, NJ: Princeton University Press.

J.M. Wing (2006). 'Computational Thinking', *Communications of the ACM,* 49/3, 33–5.

J.M. Wing (2008). 'Computational Thinking and Thinking about Computing', *Philosophical Transactions of the Royal Society*, Series A, 366, pp. 3717–25.

الخاتمة: هل يندرج علم الكمبيوتر ضمن العلوم الشاملة؟

S. Dasgupta (2014). *It Began with Babbage: The Genesis of Computer Science.* New York: Oxford University Press.

P.J. Denning & C.H. Martell (2015). *Great Principles of Computing*. Cambridge, MA: MIT Press.

P.J. Denning (2005). 'Is Computer Science Science?' *Communications of the ACM*, 48/4, 27–31.

P.J. Denning & P.A. Freeman (2009). 'Computing's Paradigm', *Communications of the ACM,* 52/12, 28–30.

P.S. Rosenbloom (2013). *On Computing: The Fourth Great Scientific Domain*. Cambridge, MA: MIT Press.

R. Snodgrass (2010). 'Ergalics: A Natural Science of Computing'. http://citeseerx.ist.psu.edu/viewdoc/download?doi= 10.1.1.180.4704&rep=rep1&type=pdf. Retrieved 16 Sept. 2015.


هنداوي